División de Ciencias Básicas e Ingeniería

Algoritmos heurísticos para el ruteo de dispositivos programables

Tesis para obtener el grado de

Maestro en Optimización
por
Fabián Guillermo Galván Cardozo

Asesores:
M. en C. Oscar Alvarado Nava.

Dr. Eduardo Rodríguez Martínez.

16 de febrero de 2016

Dedicatoria

A Oliver Nadler, por su gran pasión por la ciencia y su manera de transmitirla al mundo sin esperar nada a cambio.

Agradecimientos

Envío un especial agradecimiento a mis asesores, que dedicaron tiempo y esfuerzo para que este proyecto pudiera salir adelante con cada una de sus ideas. A mis padres y a mi hermano, por apoyarme incondicionalmente en cada una de mis ideas de vida y profesionales. A Oliver Nadler, por todas las ideas y trabajo que aportó a este proyecto.

A todos, muchas gracias.

Resumen

En este documento, se presenta una nueva manera eficiente de obtener un ruteo detallado, para cualquier tipo de FPGA con estructura de islas. A partir de instancias de ruteo global, que proporciona el programa de empaquetamiento versatil, colocación y ruteo [1], se creó un nuevo algoritmo heurístico llamado búsqueda indexada, que parte de un modelo matemático de satisfacibilidad booleana y que se resuelve mediante una coloración condicional de gráficas, que también se propone aquí. A su vez, se potencializa la heurística de búsqueda indexada, por medio de un árbol R, el cual indexa los datos de tal manera, que la heurística puede conocer rápidamente el número de conflictos, en los que aumenta o disminuye la función objetivo, si se realiza un movimiento. Con cada uno de estos aspectos, se búsca un ruteo de calidad perfecta y que además disminuya considerablemente el tiempo necesario para obtener un ruteo detallado. Finalmente, se compara el ruteo detallado de seis instancias obtenidos por el VPR y por la búsqueda indexada.

[1] VPR por sus siglas en ingles versatile place and route

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1
Introducción

1.1. Planteamiento del problema

Debido al crecimiento exponencial de la arquitectura interna de los dispositivos lógicos programables, es necesario comenzar a plantear métodos más eficientes para la colocación y el ruteo interno de estos dispositivos. Uno de los problemas más complejos de la combinatoria, pero que tiene más aplicaciones, es el ruteo, el cual está presente en nuestra vida diaria, por ejemplo, en el transporte, en el traslado de mercancía, en el diseño físico de circuitos electrónicos e incluso en los dispositivos lógicos programables, basados en arreglos de compuertas programables en campo[1].

Los dispositivos lógicos programables, inicialmente no contienen una aplicación electrónica predeterminada. Para asignarles una aplicación, el usuario describe el comportamiento electrónico que tendrá el dispositivo, mediante un lenguaje de programación y modelado para circuitos lógicos programables. A continuación, se compila la descripción para obtener una síntesis, la cual entrega como resultado un archivo con la descripción de sus respectivos bloques lógicos configurables[2], que representan la lógica del diseño e interconexiones. A partir de aquí, se realizan dos procesos, uno para asignar la posición de los bloques lógicos dentro del FPGA , i.e. el proceso colocación, y un segundo que rutea las interconexiones entre ellos, i.e. el procesos de ruteo.

Esta tesis, se centra en el segundo proceso, en el cual los bloques lógicos asignados dentro del FPGA, tienen que interconectarse mediante rutas especificadas por un ruteo global, las cuales carecen de una asignación de los recursos de interconexión internos del FPGA. El presente documento se organiza de la siguiente manera.

1.2. Organización del documento

En el Capítulo 1, se describirá el estado del arte para el ruteo de los dispositivos lógicos programables FPGA. Durante el Capítulo 2, se explica la arquitectura interna de los dispositivos FPGA. En el Capítulo 3, se describe el problema de realizar un ruteo detallado en un dispositivo FPGA. En el Capítulo 4, se explica el método de coloración condicional que se propone en esta tesis, asi como la coloración propia. En el Capítulo 5 se establecen los fundamentos básicos de la construcción de las

[1] FPGA pos sus siglas en inglés
[2] CLBs por sus siglas en ingles

gráficas de conectividad y exclusividad, que dan pie al ruteo detallado, así como el modelo matemático de satisfacibilidad booleana. En el Capítulo 6, se describe la heurística y su procedimiento de búsqueda dentro del árbol R. En el Capítulo 7, se hace una comparativa entre del ruteo detallado que proporciona el paquete VPR y la búsqueda indexada que se propone en esta tesis.

1.3. Estado del arte

Uno de los problemas más desafiantes para la combinatoria, es el problema del ruteo [13,14,17], en el cual se determinan las rutas necesarias que interconectan una o más fuentes con uno o más destinos. Este problema puede presentarse en diversos ambitos tecnológicos, de entre ellos, el más destacado es el ruteo de circuitos electrónicos. En la electrónica analógica y digital, es común que los diseños en escala micro o macro, contengan dispositivos lógicos o analógicos, que necesitan ser interconectados en una estructura de recursos limitados y que además, las rutas que se establecen en dicha estructura, jamás pueden coincidir si pertenecen a señales diferentes. En la literatura, existen distintos métodos para resolver este tipo de problemas de ruteo electrónico, que van desde la solución más simple, la cual radica en un ruteo sin asistencia de CAD, siempre y cuando el diseño electrónico sea pequeño, hasta métodos de alta complejidad como los modelos de programación lineal y entera [4] o las heurísticas [5] que buscan dar soluciones a problemas, en donde los métodos exactos no lo logran o tienen un mal desempeño. Para conocer la complejidad del problema de ruteo, se determina el tiempo que tarda una computadora en resolver el problema, para la misma instancia con diferentes entradas. Como resultado, se obtiene un polinomio, el cual indica cómo crece el tiempo de resolución del problema de decisión, de acuerdo a una entrada dada. Diversos articulos de la literatura [11] [23] sugieren tratar el problema de ruteo como un problema de satisfacibilidad[3], por lo que podemos clasificar el problema de ruteo dentro de los problemas NP-completos [6] [7].

En las últimas décadas, existe una tendencia creciente en la que los dispositivos electrónicos, comienzan a dejar de tener funciones concretas, para dar paso a los dispositivos programables. Un solo dispositivo lógico programable, puede sustituir a miles de componentes que realizan una función concreta, ya que estos dispositivos, se adaptan a los requerimientos del usuario y a su vez pueden ser programados para actualizar o cambiar la función para la que están diseñados.

Hasta ahora, uno de los mejores métodos que se han encontrado para rutear los dispositivos lógicos programables, es la técnica del buscador de caminos[4] [25]. Esta técnica se descompone en dos iteraciones principales. El primer ciclo, consiste en interconectar todos los caminos sin importar si existe congestión entre ellos, esto da como resultado una inicialización de todas las rutas. El siguiente ciclo de iteraciónes, se repite tantas veces como sea necesario y consiste en deshacer alguna ruta que tenga congestión e intentar rutearla de nuevo por otro camino. El costo de utilizar un recurso de ruteo congestionado, aumenta en cada iteración, por lo tanto, es más probable seleccionar otros caminos. Este proceso se repite, hasta que no existen rutas congestionadas.

[3] SAT
[4] pathfinder

Las ventajas de utilizar el ruteo de búsqueda de caminos, radican en que se construye una solución de cada ruta de forma iterativa, por lo cual cada red está definida en todo momento, además el ruteo inicial con congestionamientos, es la base fundamental para la construcción del ruteo de cada una de las redes. Las desventajas asociadas a esta heurística, indican que permitir rutas congestionadas, necesariamente impacta en el tiempo de ejecución del ruteo, como consecuencia del re-ruteo de cada red congestionada, además, el ruteo realizado en la primera iteración, es la base de las siguientes iteraciones, lo cual proporciona muy poca flexibilidad al ruteo.

Existen distintos tipos de dispositivos lógicos programables, entre ellos el FPGA, el cual tiene un uso comercial que va en aumento debido a su gran flexibilidad interna. Diversas compañias, realizan su propio diseño y compiten comercial y tecnológicamente para colocar su FPGA en el mercado. Uno de los modelos sobresalientes de dichas empresas, consiste en una estructura básica de filas y columnas de CLB's, con canales horizontales y verticales de interconexión que los separan [15]. Esta arquitectura llamada simétrica, servirá como base para el ruteo que se pretende realizar en este proyecto. Una característica importante del FPGA simétrico, es que cada ruta se descompone en segmentos verticales y horizontales, que sólo pueden abarcar un CLB por vez, lo cual permite modelar el ruteo de distintas formas. Durante esta tesis, se explica a detalle, tres formas distintas de modelar el problema del ruteo detallado que se resuelven con una sola heurística.

Capítulo 2

Descripción de un FPGA.

En un principio, surgieron las memorias programables de sólo lectura[1], las cuales sólo requerían de una línea de entrada de direccionamiento y una línea de salida de datos, sin embargo no era posible realizar operaciones lógicas en su interior. A partir de esto, se crearon dispositivos con arreglos de lógica programable[2] que en su interior podían realizar operaciones lógicas de tipo conjunción, disyunción y suma de productos, mediante planos de compuertas lógicas establecidas. El problema de la estructura interna de los dispositivos PLA, era que los planos de compuertas establecidas crecían rápidamente con el número de entradas. Para reducir este problema, se crearon los dispositivos lógicos programables complejos[3] y los dispositivos lógicos programables simples[4] de donde surgieron los FPGAs.

Los FPGAs son una generación de dispositivos semiconductores, que nace a partir de 1984. Desde entonces, estos dispositivos han tenido un crecimiento tecnológicamente exponencial, que ha permitido a los usuarios, investigadores y empresas, implementarlos dentro de sus diseños electrónicos digitales y analógicos. Estos dispositivos están dispuestos de tal manera, que el usuario puede programar su interconexión y su funcionalidad, mediante un lenguaje de descripción de hardware[5]. Las capacidades de estos dispositivos, pueden permitir a un diseñador programar desde pequeñas aplicaciones como una compuerta lógica, hasta grandes y complejos diseños combinacionales y secuenciales.

Los FPGAs simetricos propuestos por Xilinx en [15] , pueden representar decenas de miles de compuertas lógicas, interconectadas entre sí, debido a esto, es importante tener en cuenta el impacto que tendrá el algoritmo de ruteo sobre esta arquitectura, ya que cada año, el FPGA extiende sus capacidades de interconexión y de unidades lógicas en un cincuenta por ciento [20]. En general, en el estado del arte es posible encontrar diversos algoritmos de ruteo, como el recocido simulado, que no es escalable debido a su largo tiempo de convergencia o el método de búsqueda geometríca que resuelve los conflictos de ruta uno a la vez [19] [18].

En esta sección, se describe la arquitectura básica de un FPGA, así como sus métodos de interconectividad. En principio un FPGA está constituido por tres componentes principales, un arreglo de bloques de entradas y salidas, un arreglo de bloques lógicos configurables, y una red programable de interconexión que comunica a los elementos anteriores (Bloques S y C), tal y como lo muestra la

[1] PROM por sus siglas en ingles programmable read-only memory
[2] PLA por sus siglas en ingles programmable logic array
[3] CPLD por sus siglas en ingles omplex programmable logic device
[4] SPLD por sus siglas en ingles simple programmable logic device
[5] HDL por sus siglas en inglés Hardware Description Language

Figura 2.1. A continuación se describe ampliamente cada uno de estos tres componentes básicos en los que se basará el método de optimización para el ruteo de los FPGA.

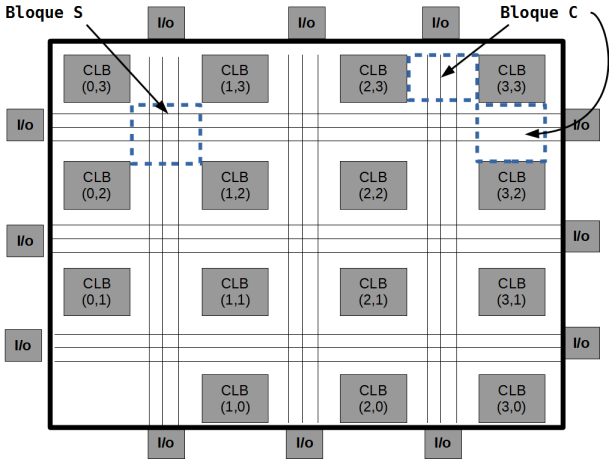

Figura 2.1: Arquitectura interna del FPGA simétrico

Bloque CLB: Un CLB es un bloque isla que está separado de sus islas vecinas por un mar de interconexiones. Este es el bloque que controla toda la lógica principal del FPGA y su función es procesar las funciones de los estados lógicos obtenidos a partir de la síntesis del codigo que se realiza en el lenguage de descripción de hardware[6]. Se compone de flip-flops D y/o T, multiplexores y tablas de consulta[7]. Su estructura interna, se divide en cuatro capas llamadas rebanadas[8]. Dos rebanadas M y dos rebanadas L [1] [9]. El SLICEL, es la unidad de procesamiento lógico del CLB y consta de tablas de búsqueda LUT y cadenas de acarreo implementadas mediante flip-flops. El SLICEM implementa espacios de memoria y funciones lógicas, como lo muestra la Figura 2.2.

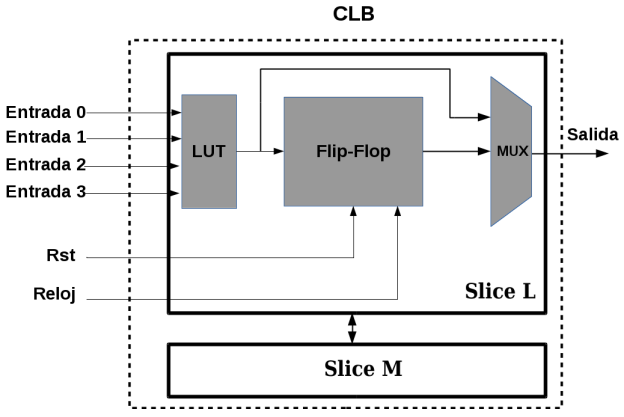

Figura 2.2: Arquitectura interna de un bloque lógico programable CLB

[6]HDL por sus siglas en ingles hardware description language
[7]LUT por sus siglas en ingles lookup table
[8]SLICE

Bloque C: El interior de un bloque C, está constituido por pistas de interconexión, las cuales abarcan longitudinalmente un bloque CLB. Estas pistas, están unidas a las de un bloque S para continuar su camino, o a las pistas de entrada-salida de un bloque CLB [9]. Su tipo de conexión depende de la arquitectura del FPGA que se utiliza en ese momento. En general, para realizar una conexión de bloque C con un CLB, la salida del CLB se conecta perpendicularmente a un segmento del canal que contiene el bloque C, mediante elementos de interconexión. Por defecto, el bloque C y el bloque S, están conectados en todos sus segmentos, esto da pie a que la señal pueda continuar su camino en la ruta deseada. Todos los segmentos que definen a una red, fozosamente atraviezan uno o más bloques C.

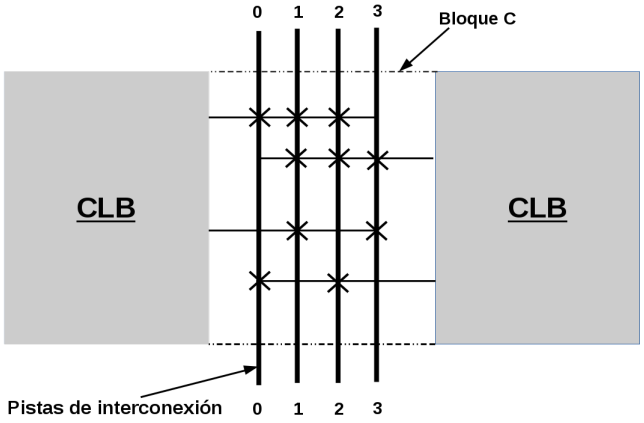

Figura 2.3: Esquema básico de un bloque C.

Bloque S: Este bloque permite dirigir la señal, de manera que esta pueda girar a 0, 90 o a 270°. Como en el bloque C, la arquitectura de conexión en este bloque varía de acuerdo al FPGA que se esté utilizando. En la Figura 2.4, se muestran dos posibles arquitecturas de bloque S, por las que una señal puede viajar [9].

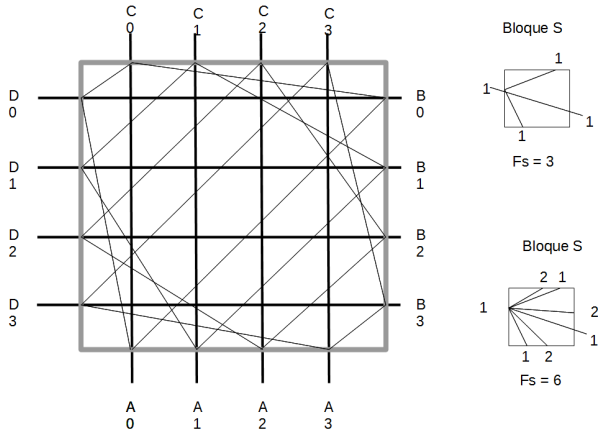

Figura 2.4: Esquema básico de las conexiones en un bloque S y dos arquitecturas posibles $Fs = 3$ y $Fs = 6$.

Para definir el tipo de FPGA que se está utilizando además de los tres bloques mencionados anteriormente, se deben de agregar tres tipos de parámetros:

W.- Indica el número de pistas que atraviesan los canales horizontales/verticales.

Fc.- El número de pistas con las que puede conectarse un bloque CLB con un bloque C.

Fs.- Indica al bloque S las interconexiones que se puedan realizar para una entrada dada.

Para este problema de optimización se utiliza una W variable de acuerdo al problema, una Fc= W y un Fs=3. En la siguiente Figura puede observarse gráficamente la representación de estos parámetros.

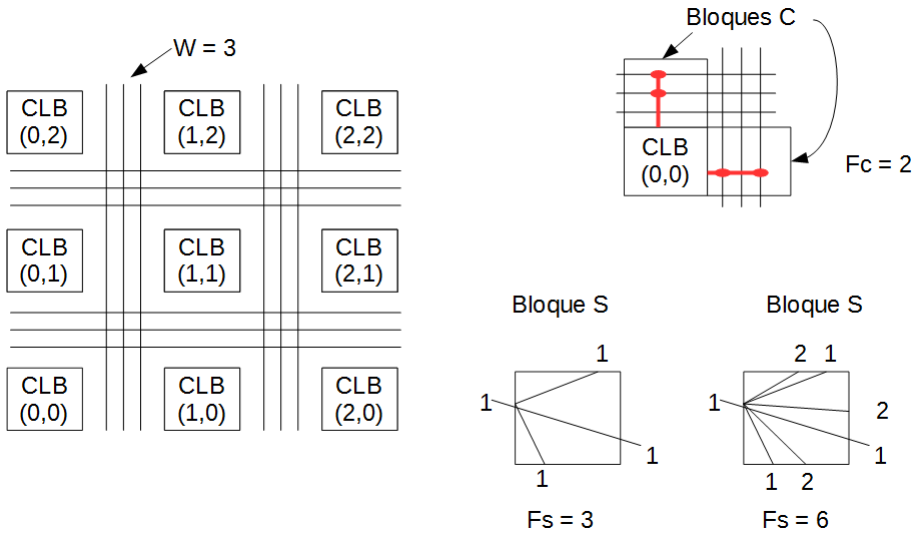

Figura 2.5: Parámetros necesarios para la elección de arquitectura del FPGA. W, Fc, Fs.

Etapas de Programación de un FPGA.

Como se mencionó anteriormente, un FPGA por defecto no contiene una aplicación lógica para desempeñar sus funciones. por lo tanto, la primera tarea que tendría un programador de FPGA's, es asignar funciones lógicas a los bloques CLB, así como describir las entradas y salidas que dan pie a estas funciones, además es necesario distribuirlos espacialmente dentro del FPGA. Esta sería una tarea demasiado compleja para un ser humano, ya que describir un circuito a través de funciones lógicas, requiere un alto conocimiento de síntesis lógica, por ello existen herramientas que facilitan la síntesis, colocación y ruteo de estos dispositivos. Las tres etapas que a continuación se muestran, describen el proceso mediante el cual un FPGA es programado para realizar alguna función.

Descripción de hardware: Durante esta etapa, el programador utiliza un lenguaje de programación, llamado lenguage de descripción de hardware[9], en el cual describe el funcionamiento de cada uno de los bloques del circuito digital que se desea implementar en el FPGA.

[9]HDL por sus siglas en inglés hardware description language

Síntesis y Colocación: Una vez realizado el código en HDL, este se sintetiza y se convierte en funciones lógicas, que serán asignadas a cada uno de los bloques CLB dispuestos internamente en el FPGA. La asignación de estos bloques CLB, no puede ser aleatoria y se realiza mediante una técnica de recocido simulado [10], que intenta reducir la distancia de interconexion entre bloques CLB y/o salidas-entradas que comparte una señal. La colocación propuesta impactará de forma directa al problema del ruteo.

Ruteo: Una vez que se han colocado todos los bloques CLB, el ruteo comienza a generar interconexiones entre pines que contengan la misma señal. Estas interconexiones, respetan ciertas reglas básicas, como por ejemplo, evitar que un canal de capacidad n pistas, contenga $n+1$ redes o que dos señales viajen sobre una misma pista. Es por ello, que para facilitar este trabajo, el ruteo se descompone en un ruteo global y en un ruteo detallado.

Ruteo global: Dada una disposición de los bloques CLB, el ruteo global intenta en cada iteración, encontrar una ruta de interconexión entre bloques CLB y entradas-salidas que contengan la misma señal. Como objetivo, cada ruta intenta construirse con la menor distancia posible, esto evita que se utilicen recursos de interconexión que pueden aprovechar otras redes (Menor congestionamiento). La ventaja que tiene este ruteo, radica en dejar inicializadas las rutas, para que el ruteo detallado sólo tenga que asignar pistas a los segmentos que componen la ruta. Cualquier ruta generada por el ruteo global, necesariamente hace usos de los segmentos de bloque S y C que se describieron anteriormente.

Ruteo detallado: Consiste en tomar las rutas proporcionadas por el ruteo global y asignarles a cada una, pistas de bloque S y de bloque C, de tal manera que sus segmentos no se mezclen con segmentos de otras redes y que el camino de toda la red, esté completamente conectado.

2.1. Motivación

En la literatura, existen distintos algoritmos que intentan realizar un ruteo detallado, i.e. algoritmo gloton, Dykstra y borde izquierdo, sin embargo, estos algoritmos no garantizan resultados y tiempos de ejecución de buena calidad [8] [2] [3]. Por ello, en esta tesis, se creó una nueva metodología, que incluye dos restricciones, las cuales permiten manejar el modelo de una manera práctica. La restricción de conectividad permite asegurar, que las conexiones punto a punto establecidas por el ruteo global estarán completamente interconectadas a través de secuencias de bloques C y S. Para fines de la construcción de la ecuación de satisfacibilidad, cada conexión punto a punto, implica una restricción de conectividad. La restricción de exclusividad asegura que los segmentos de red que atraviesan cada uno de los bloques C, tengan asignada una pista diferente si pertenecen a señales distintas. Por cada bloque C en donde existan dos o más segmentos, se añade una restricción de exclusividad a la función de satisfacibilidad booleana. Estas dos restricciones se resuelven mediante una coloración propia, y una nueva coloración condicional, respectivamente. Dichas coloraciones, satisfacen conectividad y exclusividad por medio de un algoritmo que basa su principio en el árbol R [12]. En este proyecto, el ruteo detallado se descompone en tres módulos principales. El primer módulo que adquiere y procesa los datos del ruteo global, transformandolos en gráficas, matrices y funciones, que son utilizadas por el segundo módulo (heurística) para resolver el ruteo. El tercer módulo recaba los datos del ruteo detallado y los preprocesa para que puedan ser comprendidos por el humano. En los siguientes capítulos,

se explica detalladamente el modelo a seguir, la estructura de las gráficas de exclusividad y conectividad, la coloración propia utilizada para generar la conectividad y el método de coloración condicional propuesto en esta tesis, el cual da pie a las condiciones de exclusividad.

Capítulo 3

Descripción del problema

Para asignar una aplicación al dispositivo FPGA, el diseño realizado en un HDL dede de atravesar por tres etapas básicas, una síntesis, una colocación y un ruteo. Para realizar la etapa ruteo, es necesario que la síntesis y la colocación se hayan realizado previamente, lo cual da como resultado, bloques CLB colocados de manera específica dentro del FPGA, con interconexiones virtuales (sin asignación de pistas) entre ellos. El ruteo de estas interconexiones virtuales, puede separarse en dos tipos de ruteo. Un ruteo global que mediante una técnica de ruteo por laberinto [25], establece caminos disponibles dentro del FPGA para las interconexiones virtuales, sin asignar pistas físicas. Y un ruteo detallado, que será objeto de estudio de esta tesis, el cual asigna las interconexiones de los caminos virtuales a pistas físicas del FPGA, cuidando siempre que la ruta esté completamente conectada y que dos conexiones que contengan señales diferentes no ocupen la misma pista. Para realizar un ruteo detallado, es necesario que el ruteo global se haya completado previamente, lo cual da como resultado, un archivo en donde se describen los caminos que toman las redes de interconexión entre CLB's.

Para poder resolver el problema de ruteo detallado, el archivo de ruteo global se codifica a un problema de satisfacibilidad Booleana, que se encuentra restringido por dos condiciones, una restricción de conectividad, que asegura que cada red este completamente conectada y una restricción de exclusividad que permite evitar que dos redes ocupen la misma pista física. Para resolver el problema de satisfacibilidad booleana mencionado anteriormente, se generan dos tipos de gráficas que reflejan el nivel de conectividad y exclusividad en la fórmula booleana SAT. Por lo tanto, al resolver la coloración de estas gráficas, implícitamente se está resolviendo el problema SAT. Resolver la coloración de estas gráficas, tiene una complejidad NP-completo [6] [7], lo cual implica utilizar una heurística que base su comportamiento en la ubicación espacial de los datos de las gráficas mediante un árbol R.

3.1. Formato de entrada

Como entrada para este problema de optimización, se tiene un archivo de ruteo global, el cual indica el camino por el cual se conecta un CLB con otro u otros CLBs, o con una o más salidas-entradas. Para describir este ruteo, el archivo de ruteo global, descompone cada red en segmentos horizontales llamados segmentos X y verticales llamados segmentos Y , los cuales abarcan longitudinalmente un bloque CLB. A continuación, en el Cuadro 3.1, se muestra un ejemplo de un archivo de entrada de ruteo global y se define la funcionalidad de cada abreviatura que este archivo contiene.

W0,0,0	W0,1,3	W3,0,3
Y0,0	X0,1	X3,0
Y0,1	X1,1	Y2,1
X1,1	Y1,2	X2,1
Y1,2	X2,1	W2,2,1
X2,2	Y2,2	X2,1
X3,2	Y2,1	X1,1
Z3,3,1	Z3,1,2	Y0,2
		Y0,3
		Z0,3,0

Cuadro 3.1: Entrada del problema de optimización.- Ejemplo de ruteo global de tres redes

W: Indica el comienzo de una señal, o el final de ella, en un CLB o en una entrada-salida. Contiene tres columnas que dan referencia a las coordenadas del CLB y el pin al que se conecta la señal.
X: Denota un segmento horizontal que contiene las coordenadas (x, y) del CLB al que pertenece dicho segmento.
Y: Denota un segmento vertical que contiene las coordenadas (x, y) del CLB al que pertenece dicho segmento.
V: Significa que hay un corte de adyacencia, es decir un cambio de señal.
Z: Tiene la misma función que W, pero se utiliza al final de una red para indicar que esta no contiene más caminos.

El archivo mostrado en el Cuadro 3.1 se interpreta como sigue. Para poder colocar gráficamente un segmento de red, se comienza leyendo el tipo de segmento al que pertenece y a continuación se leen las coordenadas. Si el segmento tiene la leyenda X (horizontal), entonces se ubica el CLB con las coordenadas obtenidas y se coloca el segmento horizontal en la parte superior del CLB. Si el segmento tiene la leyenda Y (vertical), entonces se ubica el CLB con las coordenadas obtenidas y se coloca el segmento vertical a la derecha del CLB. En la Figura 3.1 se muestra el ruteo global generado por el Cuadro 3.1

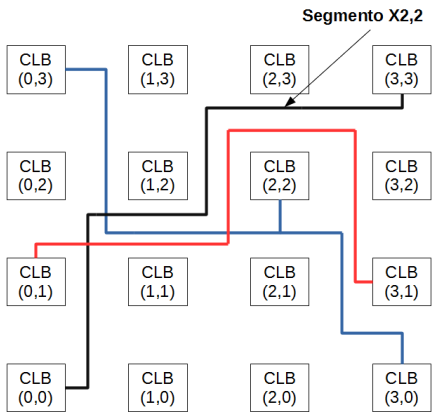

Figura 3.1: Gráfico del ruteo global de la tabla 3.1

3.1.1. Usos de la notación del ruteo global.

El ruteo global de una instancia, puede tener distintas características para cada red que contiene, por ello, mediante el Cuadro 3.2 se explican las posibles notaciones que pueden codificar el trayecto de una red.

Red 1	Red 2	Red 3
W0,0,1	W0,0,3	W0,0
X0,0	X0,0	X0,0
X2,0	VX1,0	X1,0
Z2,0	Z1,0,2	X2,0
		Z2,0,1

Cuadro 3.2: Notaciones del ruteo global

La red 1 del Cuadro 3.2, tiene una configuración que parte de dos segmentos (X0,0 y X2,0) separados al menos por dos canales y por lo tanto, no pueden tener adyacencia debido a que no hay un bloque S que los conecte en ese orden. La conectividad de los dos segmentos de red en esta configuración, es trunca.

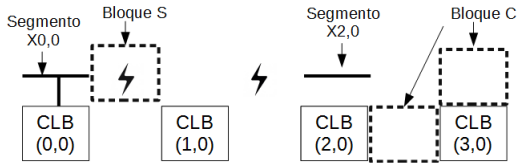

Figura 3.2: Ruta que no permite una conexión.

En la red 2 del Cuadro 3.2, si hay un bloque S que interconecta a X0,0 y X1,0, pero no puede existir una adyacencia, debido a que 'V' indica que hay un corte de adyacencia.

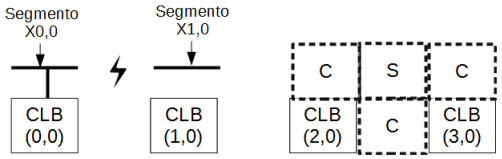

Figura 3.3: Ruta en la que la no adyacencia separa las señales

En la red 3 del Cuadro 3.2, si se permite adyacencia debido a que hay un bloque S que los interconecta en ese orden, y por lo tanto, la conectividad de estos dos segmentos es perfecta. En general, esta configuración es la que predomina en el ruteo global, ya que lo que se busca es que cada segmento que describe la ruta de una red, sea una conexión con su siguiente segmento.

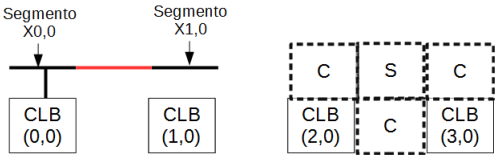

Figura 3.4: Ruta con conectividad perfecta

3.2. Estructura de datos

En esta sección, se describe el procesamiento de los datos obtenidos a partir del archivo del ruteo global. La lectura de estos datos, puede realizarse mediante una sola columna, sin embargo los datos también podrían estar organizados en dos o más columnas. Incluso es posible definir todos los segmentos y todas las redes en una sola línea. El módulo de lectura, necesariamente debe de considerar como red, sólo aquellas que usan al menos un bloque C, i.e. un segmento X o Y, sino es el caso, entonces la red es ignorada. Por ejemplo, el Cuadro 3.3 muestra una descripción de ruteo que no contiene segmentos x o y, por lo tanto no hay un camino para esta red.

W0, 0, 0
W0, 0, 0
W1, 0, 0
Z 2, 0, 0

Cuadro 3.3: Red ignorada por no contener un bloque C

Una red está constituida por una o varias subredes que interconectan una fuente con uno o más destinos. Para facilitar el problema del ruteo, se establece que la subredes que conforman una red, están definidas por un camino de punto a punto. La Figura 3.5, muestra un ejemplo de una red que contiene tres subredes.

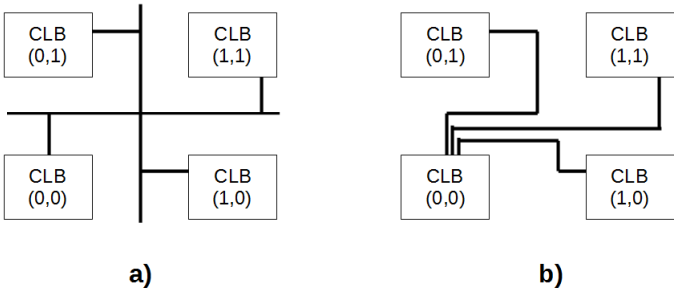

Figura 3.5: a) Representación de una red completa. b) Descomposición de la red en subredes de punto a punto.

3.2.1. Parámetros de caracterización del FPGA

Una vez que se ha leido la estructura del ruteo global, se procede a leer los siguientes valores que caracterizan la estructura del FPGA para un problema dado:

X mínima, X máxima, Y mínima y Y máxima. Estos cuatro datos nos indican las coordenadas en el espacio bidimensional en donde existen las redes y se obtienen por medio de la lectura de los datos del archivo de ruteo global. Al conocer estos datos, el usuario puede saber con claridad si el FPGA que está utilizando es el correcto en cuanto al espacio del ruteo o si es necesario cambiar de arquitectura. En todo momento estas cotas deberán ser respetadas por el ruteo detallado. En el Cuadro3.4, se hace referencia a estos valores por medio de las variables $xMin$, $xMax$, $yMin$ y $yMax$.

Número de redes existentes en el circuito. El módulo de lectura obtiene la cantidad de redes existentes en el circuito, contando el número de veces en los que la sigla Z aparece en el ruteo global. Este cálculo, sirve al programa para verificar que la instancia ha sido leída correctamente.

Número de vértices C. Para calcular este número, es necesario contar la cantidad de veces que aparece un segmento X o Y en el ruteo global. Este número es necesario para generar la matriz que aloja las gráficas a colorear. El vértice C, indica que un segmento atravieza un bloque C.

Por cada W y Z se suman seis vértices C y se agrega un vértice L. El vértice L indica que hay un pin que conecta el CLB con un bloque C.

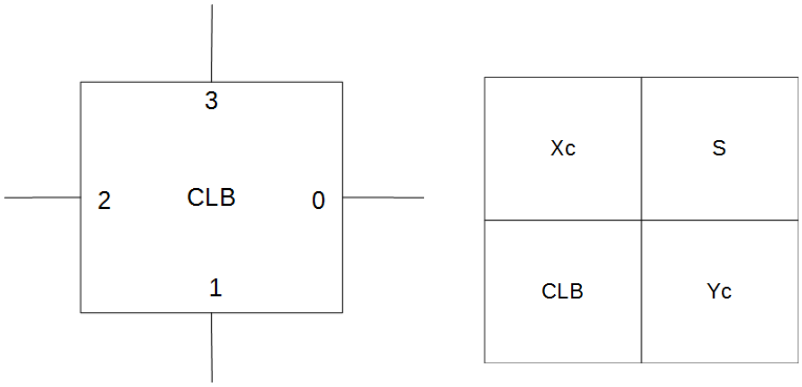

Figura 3.6: a) Número mínimo de pines, b) Paquete: Consta de un CLB, un bloque S y dos bloques C

Ecuación	Descripción	Valores para el Cuadro 3.1
$X = xMax - xMin + 1$	**Tamaño de bloques X:** Describe el espacio horizontal medido en bloques C, que abarca el ruteo.	$3 - 0 + 1 = 4$
$Y = yMax - yMin + 1$	**Tamaño de bloques Y:** Describe el espacio horizontal medido en bloques C, que abarca el ruteo.	$3 - 0 + 1 = 4$
$minPin = zMax - zMin + 1$	**Número mínimo de pines por CLB:** Indica cuántos pines debe de tener cada uno de los CLB's, como se muestra en la Figura 3.6a.	$3 - 0 + 1 = 4$
$\#paq = X * Y$	**Número de paquetes:** Un paquete está constituido por un bloque CLB, un bloque C horizontal, un bloque C vertical y un bloque S, como lo indica la Figura 3.6b. La fórmula, indica cuantos paquetes son necesarios para una instancia dada.	$4 * 4 = 16$
$\#pines = \#paq * zSize$	**Número total de pines utilizados:** Calcula el número de pines que utilizan todas las redes de la instancia.	$16 * 4 = 64$

Cuadro 3.4: Ecuaciónes de los parámetros de caracterización del FPGA

3.3. Modelo del problema.

Cuando el módulo de lectura ha obtenido todos los datos anteriores, se realiza un análisis de interconectividad entre subredes y se establece la estructura para generar las gráficas que restringen por conectividad y exclusividad las pistas de cada red. El análisis de interconectividad se basa en la teoría de conjuntos. Cada conjunto se encuentra relacionado a los segmentos de una sola red. Las intersecciones que se generan en estos conjuntos de redes, indican que dichas redes comparten camino en algún punto de su trayectoria. En la Figura 3.7, se muestra un ejemplo de los conjuntos obtenidos a partir de la instancia mostrada en el Cuadro 3.1. Las intersecciones de estos conjuntos, indican que los segmentos que se encuentran dentro comparten el mismo bloque C y por lo tanto pueden generar problemas en el ruteo.

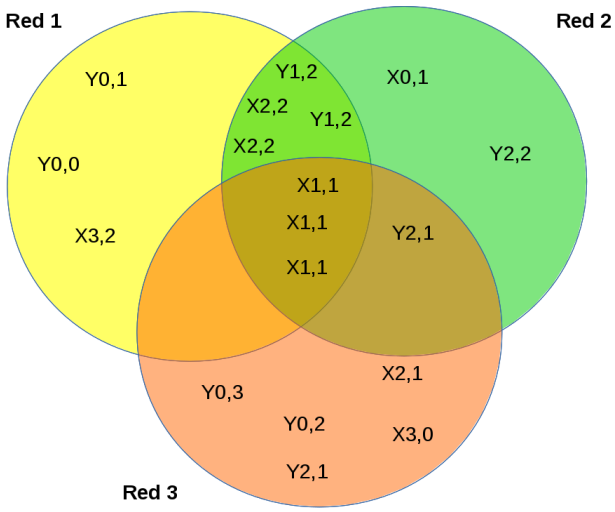

Figura 3.7: Conjunto conformado por tres redes de la instancia en estudio.

Para solucionar el problema del ruteo detallado, son necesarias dos restricciones:

Exclusividad. En cualquier circuito electrónico es necesario evitar que dos señales ocupen la misma pista, ya que esto puede producir un corto circuito o grandes alteraciones en la información que conlleva esa señal. Por ello es necesario que esta restricción se ocupe de evitar dichos acoplamientos a menos que estos provengan de la misma señal. En caso de que exista un acoplamiento de dos señales distintas, la función de esta restricción será indicar que existe un error de corto-circuito y que el ruteo debe de ser modificado hasta que no exista ningún tipo de acoplamiento entre señales diferentes.

Conectividad. Esta restricción, asegura que una red esté completamente conectada, por cada uno de sus segmentos por medio de un camino de bloques C y S. Lo cual asegura que una señal viaje correctamente desde su destino hasta su fuente.

La Ecuación(3.1), representa la función objetivo para el problema de asignación, considerando las restricciones previamente descritas. El primer termino de conjunción modela la restricción de conectividad, mientras que el segundo corresponde a la restricción de exclusividad.

$$\text{Maximizar Args f} = \sum_{r\in R}\sum_{i\in S}\delta(X_{(r,i)} = X_{(r,i+1)}) + \sum_{k\in Bc}\sum_{j\in T}\delta(Y_{(k,j)} = Y_{(k,j+1)})$$

$$\text{sujeto a:} \tag{3.1}$$

$$0 \leq X_{r,i} \leq C$$
$$0 \leq Y_{k,j} \leq C$$

Donde $X_{r,i}$ es la pista asignada al segmento i del conjunto de pistas C de la red $r \in R$, R es el conjunto de redes, i es un segmento de la red r, $Y_{k,j}$ es la pista asignada al segmento $j \in C$, para un bloque C $k \in Bc$, Bc es el conjunto de bloques C, y $\delta(m) = 1$ si la condición m se cumple.

Durante el modelado de este problema de optimización se utilizaron tres formas distintas de realizar el ruteo.

Ruteo por segmentos. En este tipo de ruteo se intenta asignar a cada segmento una pista, que puede ser diferente a la de su segmento contiguo. La única restricción que este modelo requiere, es que dos segmentos de dos redes diferentes con las mismas coordenadas no ocupen la misma pista, a esta restricción se le conoce como de exclusividad.

Ruteo por subredes. Una red completa, parte de una fuente (CLB o entrada-salida) y puede tener n destinos. Para facilitar el modelado de este tipo de ruteo, esta red se descompone en n caminos que sólo contienen una fuente y un destino. A cada uno de los n caminos se les asigna una pista. Cabe destacar que en este modelo existen dos restricciones, una de conectividad, que asegura que la red esté completamente conectada y una de exclusividad que asegura que dos señales de diferentes redes no ocupen una misma pista.

Ruteo por redes. Para este tipo de modelo a cada red completa se le asigna una sola pista, es decir, durante todo el trayecto de esta red, sólo podrá ocupar una pista en todos sus segmentos para llegar a todos sus destinos. La única restricción utilizada, es la de exclusividad que al igual que el ruteo por subredes, asegura que dos señales de diferentes redes no ocupen una misma pista.
Para asegurar conectividad se toma como base el recorrido que proporciona el ruteo global y se revisa que en ningún punto de este recorrido exista una desconexión. En caso de que existiera alguna desconexión, la función de esta restricción será indicar que existe un error y por lo tanto el ruteo deberá llevarse a cabo nuevamente hasta que no exista ninguna desconexión.

3.4. Satisfacibilidad Booleana.

La satisfacibilidad booleana (SAT), es un problema que consiste en determinar la asignación de n variables para que una función booleana f sea verdadera en toda su expresión. Si alguna de estas variables no cumple con la función SAT, se dice que la asignación no es óptima. Por lo general los programas utilizados para resolver una instancia SAT, modelan cada instancia mediante ecuaciones conocidas como formas normales conjuntivas[1]. Cada ecuación CNF, consiste en una o más clausulas

[1] CNF por sus siglas en ingles conjunctive normal form

de disyunciones unidas por conjunciones. Cualquier función booleana puede transformarse a su forma normal conjuntiva.

Para resolver el problema del ruteo detallado, el método de satisfacibilidad booleana, transforma el espacio de islas del FPGA en una gran función atómica y booleana. Para lograr construir esta función, es necesario tener una instancia que describe completamente el ruteo global, la cual se codifica a un problema SAT, por medio de cláusulas que evaluan la conectividad y exclusividad de cada ruta del archivo. Para conformar una cláusula de conectividad, se toman en pares los segmentos que definen una red del archivo de ruteo global y cada par representa una cláusula que indica si los dos segmentos cumplen con la función NXOR $\rightarrow 1$. Para conformar una cláusula de exclusividad, se ubican todos los bloques C, en donde exista más de un segmento. Los segmentos contenidos en cada uno de estos bloques C, representarán una cláusula que indica si los segmentos cumplen con la función XOR $\rightarrow 1$. Debido a los requerimientos del ruteo, el problema de optimización que aquí se estudia, sólo termina cuando el ruteo es perfecto, es decir cuando la conectividad es completa y la exclusividad indica que no hay cortocircuito en cada una de las redes. Si la función no se satisface, podemos estar seguros de que no es posible rutear el problema con esa arquitectura de FPGA.

Capítulo 4

Coloraciones

La teoría de gráficas, es una rama de las matemáticas y de las ciencias de la computación, que surge en el siglo XVIII con el problema de los puentes de Konigsber. Este problema, intentaba encontrar un camino para recorrer siete puentes existentes en el rio Pregel, de tal manera que todos los puentes se recorrieran una sola vez. En general, la teoría de gráficas se encarga de analizar las propiedades cualitativas y cuantitativas de las gráficas. La estructura de una gráfica está compuesta en su forma más básica de un conjunto de puntos o vértices y un conjunto de líneas llamadas aristas que interconectan a los vértices. Actualmente las gráficas tienen aplicaciones en distintas ramas, tales como la combinatoria, las ciencias de la computación, la topología, entre otras. En particular la teoría de gráficas ha tenido un gran impacto en la electrónica digital y analógica, ya que a medida que los componentes utilizados para realizar un diseño electrónico aumentan, intentar obtener las rutas que los interconectan se vuelve una tarea complicada. De igual forma, los dispositivos lógicos programables aumentan el doble de su capacidad año con año y por lo tanto, las rutas de interconexión en un medio limitado se vuelven más complejas. Gran parte del desarrollo de esta tesis, se basa en la teoría de gráficas, la coloración de las mismas, y en la indexación de datos mediante árboles, todo esto con el fin de conseguir un ruteo detallado que se ajusta perfectamente a la arquitectura del dispositivo FPGA que se esté utilizando.

En este capítulo, se describen dos tipos de coloraciones de gráficas para el ruteo detallado, así como su proceso de construcción y resolución. Con base en la descripción del ruteo global, se construyen dos gráficas, las cuales serán descritas en el Capítulo 5. Estas gráficas, constituyen dos restricciones que interactúan una con la otra, para solucionar el problema de satisfacibilidad booleana. La primera restricción de conectividad, obliga a los vértices de una red a estar conectados en toda su trayectoria, lo cual es posible resolver mediante una coloración condicional, que se propone en esta tesis. La segunda restricción de exclusividad, asegura que ningún segmento de la red, comparte la misma pista en un mismo bloque, a menos que sean de la misma señal. Durante la resolución de la satisfacibilidad booleana, a cada vértice se le asigna un color de acuerdo a las aristas de exclusividad y/o de conectividad que éste contenga y sólo permanece con el color seleccionado si cumple con las dos coloraciones a la vez, siempre y cuando contenga una o más aristas de exclusividad y una o más aristas de conectividad. En el caso de que sólo contenga una arista, se resuelve el problema aplicando la coloración que corresponda a dicha arista, una arista de conectividad implica coloración condicional y una arista de exclusividad implica coloración propia. En caso de no contener ninguna arista, el vértice puede tomar cualquier color, por lo tanto este tipo de casos, no se toman en cuenta. Si alguna de las dos coloraciones (restricciones) no se cumplen en todos los bloques de la red, el ruteo queda trunco o en cortocircuito.

4.1. Coloración condicional.

Dada una grafica $G = (V, E)$, donde V es un conjunto cuyos elementos son denominados vértices o nodos, E un subconjunto de pares no ordenados de vértices que reciben el nombre de aristas o arcos y F una función de restricción asignada a cada arco $e \in E$. El grado $d(v)$ de cada vértice $v \in V$, es el número de incidencias en v. Una coloración condicional, mapea cada vértice v a un color $c \in C$, donde C es el conjunto de colores a utilizar, tal que la función de restricción F que se encuentra en el arco $e \in E$, se cumple para los vértices del arco.

A diferencia de una coloración propia, que intenta colorear los vértices de tal manera que dos vértices que comparten una arista no contengan el mismo color, una coloración condicional restringe por medio de las aristas, la manera en la que se asigna colores a los vértices. Las funciones de restricción en cada arista son modeladas como tablas de verdad booleanas, de forma que los colores permitidos para los vértices de la arista, son sólo aquellos que aseguran que la evaluación de la función de restricción sera verdadera. La coloración condicional, puede asignar restricciones distintas a cada arista de la gráfica G, en esta tesis, se asignan dos tipos de restricciones a las aristas, una XOR $\rightarrow 1$ para la exclusividad y una NXOR $\rightarrow 1$ para la conectividad. La Figura 4.1 muestra un ejemplo de una coloración propia de una gráfica que tiene como restricción entre aristas una función NXOR y que sólo es óptima cuando la función toma valores verdaderos.

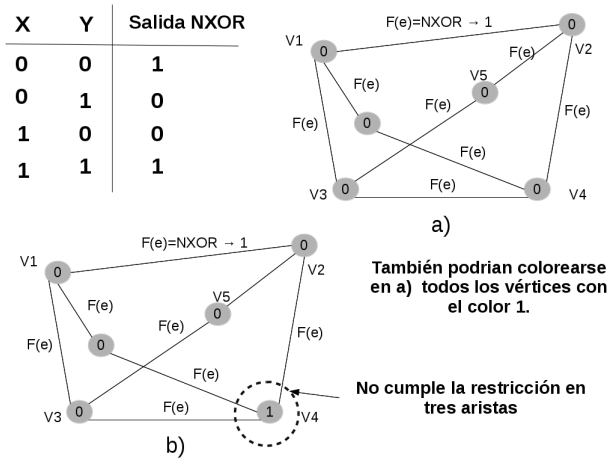

Figura 4.1: a) Gráfica que cumple con la restricción NXOR $\rightarrow 1$. b) Gráfica que no cumple con la restricción NXOR $\rightarrow 1$.

En particular, la coloración condicional que se aplica para el ruteo detallado del FPGA, tiene como principio de restricción la arquitectura del bloque S que se esté utilizando. Para una arquitectura de bloque S con Fs = 3, las restricción de conectividad de cada arista para la coloración condicional, indica que los vértices que están unidos por esa arista necesariamente tienen que ser del mismo color, ya que un color diferente indicaría una desconexión en la ruta que sigue la señal. Para una arquitectura de bloque S, con Fs=6, la restricción de conectividad facilita en gran manera el ruteo, ya que para una entrada dada se tienen dos opciones de salida. Esto significa, para una arista dada, que cuando se fija el color en un vértice, el otro vértice tiene dos opciones para colorearse. Finalmente la arquitectura de

bloque S que para una entrada permite conectarse en su salida con cualquier pista, la restricción de conectividad es exactamente igual a la restricción de una coloración propia. Por lo tanto una coloración propia es un caso particular de una coloración condicional.

4.2. Coloración propia

Una coloración propia de $G = (V, E)$, es un mapeo, $c : V \Rightarrow C = (1, ..., j)$, tal que $c(v) \neq c(w)$ si v y w son adyacentes en G. En caso de que exista solución a este mapeo, se dice que G tiene una k-coloración. Cuando k es el entero mínimo dentro de todas las soluciones factibles del problema de coloración, se le llama número cromático de G, y se denota por $\chi(G)$.

El siguiente, es un modelo lineal, en donde dada una gráfica $G = (V, E)$ y un conjunto de colores C. Se escoge una variable de decisión $X_{i,j}$ que sólo toma valores verdaderos si el vértice i es coloreado con el color j. El modelo lineal de la ecuación 4.1, tiene como función objetivo minimizar el número de colores utilizados en una gráfica para colorear propiamente.

$$\text{Minimizar g} = \sum_i \sum_j j * X_{i,j} \text{sujeto a:} \sum_j X_{i,j} = 1, \forall i \in V X_{i,j} + X_{i+1,j} \leq 1, X_{i,j} \in \{0, 1\}, \forall i \in V, \forall j \in C \tag{4.1}$$

Las restricciones con las que cuenta este modelo indican a la gráfica que es esencial que cada vértice contenga un color, además asegura que los vértices que sean adyacentes contengan colores diferentes y finalmente indica a la variable de decisión que se encuentra acotada entre valores cero y uno.

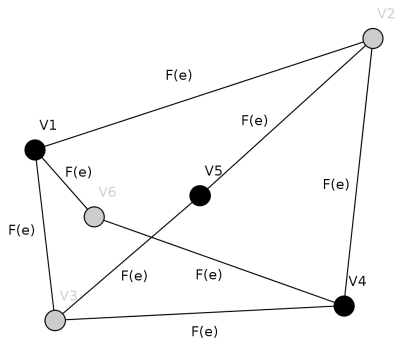

Figura 4.2: Coloración propia

4.3. Coloración propia mediante coloración condicional

Dada una gráfica $G = (V, E)$, es posible asignar una función de restricción $F(e)$ a cada elemento de E, tal que los vértices conectados a la arista e sean distintos, tal y como se muestra en la Figura 4.2. La función de restricción $F(e) = c(V_i) \neq c(V_j)$ para la coloración propia, es la función de restricción de la arista $e = (Vi, Vj)$, y c es el mapeo que asigna un color al vértice n. Dicha función $F(e)$, indica que

dos vértices adyacentes, no pueden tener el mismo color (Coloración propia), de lo contrario, existirá un conflicto. Esto se muestra en la Figura 4.3.

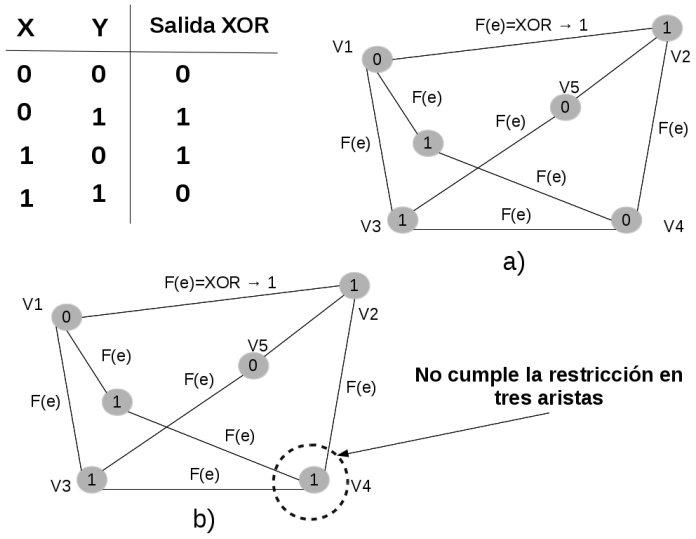

Figura 4.3: Coloración propia mediante coloración condicional.

Capítulo 5

Modelo de gráficas.

A partir de la representación física del ruteo global, se construyen dos gráficas que modelarán a la función de satisfacibilidad booleana. Estas gráficas, describen completamente el comportamiento de todas sus redes y restringen perfectamente las condiciones de conectividad y exclusividad. Es importante hacer notar que para resolver este problema, los colores asignados a la gráfica deben de cumplir con dos tipos de coloraciones, una coloración propia y una coloración condicional. Si algún color impuesto a la gráfica resultante no satisface las dos coloraciones a la vez, algunos de los caminos especificados por el ruteo global quedarán disconexos o estarán en cortocircuito. El diseño, construcción y comportamiento de estas gráficas, se detallará mediante el ejemplo mostrado en la Figura 5.1, la cual muestra gráficamente el ruteo global del Cuadro 3.1.

La construcción de las gráficas, comienza numerando cada uno de los segmentos que se encuentran en el archivo de ruteo global. Esta numeración corresponderá exactamente con el orden de aparición de los segmentos. Una vez numerados todos los segmentos de todas las redes, la heurística comienza una búsqueda exhaustiva para ubicar los bloques C que contengan dos o más segmentos de red, ya que en estos bloques se concentran todos los conflictos de exclusividad que el ruteo detallado puede contener. La Figura 5.1, muestra los segmentos numerados y la ubicación de los bloques C que tienen probabilidad de contener un conflicto para el ejemplo descrito en el Cuadro 3.1.

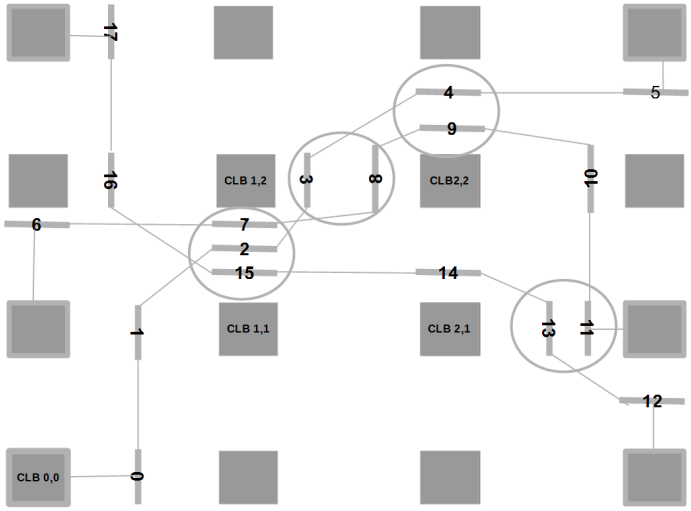

Figura 5.1: Disposición del ruteo global, numeración de segmentos y ubicación de conflictos

La gráfica de conectividad, tiene como vértices a los segmentos que conforman las redes. Estos vértices se unen mediante una arista, si sus segmentos correspondientes pertenecen a la misma red. La coloración óptima de esta gráfica, depende del nivel de ruteo seleccionado, ya que para cada nivel existen funciones de restricción diferentes. Estos niveles de ruteo, se definirán más adelante. Para cualquier gráfica de conectividad, tanto el número de colores como la función de restricción dependen de la arquitectura seleccionada. El número de colores será igual a W (i.e. el número de pistas por canal), y la función de restricción dependerá de la arquitectura del bloque S, definida por la variable Fs, como se muestra en la Ecuación 5.1. La Figura 5.2, muestra la gráfica de conectividad obtenida a partir del ruteo global de la Figura 5.1.

$$f(e) = \begin{cases} C(u)\,NXOR\,C(w) & \text{si } Fs = 3, \\ (C(u)\,NXOR\,C(w))\,OR\,(C(u)\,NXOR\,C(w)+1) & \text{si } Fs = 6, \\ 1 & \text{si } Fs = W^3. \end{cases} \tag{5.1}$$

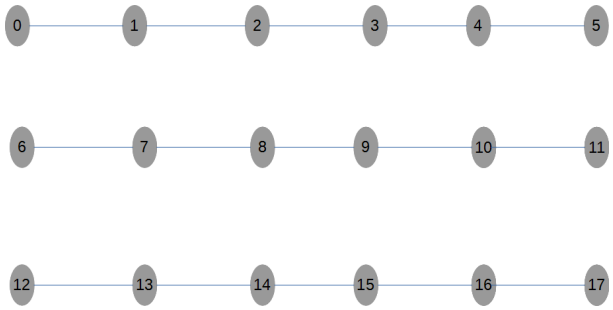

Figura 5.2: Gráfica de conectividad

La gráfica de exclusividad se construye a partir de los bloques C con más de dos segmentos. Cada segmento forma un vértice en la gráfica, los cuales se unen con aristas de exclusividad. Una arista de exclusividad difiere de una arista de conectividad, en que no tiene una función de restricción asociada. La gráfica de exclusividad sólo será una gráfica completa si el CLB se conecta con todas las pistas del bloque C. En este caso, la asignación de pistas a cada segmento se resolverá mediante una coloración propia. En caso contrario, es necesario usar una coloración condicional,en donde la restricción que tengan las aristas, depende de la conexión del bloque CLB con el bloque C. En esta tesis, sólo se considera instancias de ruteo global que caen en el primer caso. La Figura 5.3, muestra la gráfica de exclusividad construida a partir de los bloques C en conflicto, para el ejemplo de la Figura 5.1.

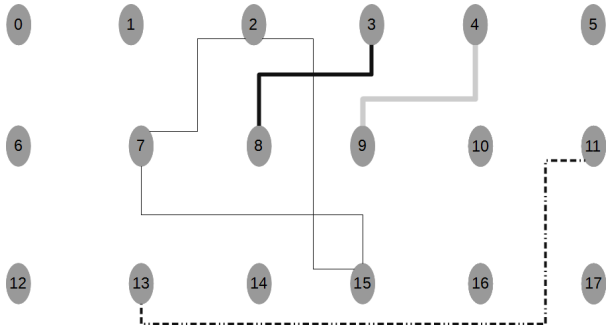

Figura 5.3: Gráfica de exclusividad

Finalmente, se concentran las dos gráficas en una sola, para permitir a la heurística evaluar las dos restricciones de conectividad y exclusividad al mismo tiempo. A esta gráfica le llamaremos gráfica de conectividad-exclusividad. La Figura 5.4, muestra la unión de las dos gráficas que sirven a la heurística de base para comenzar el ruteo.

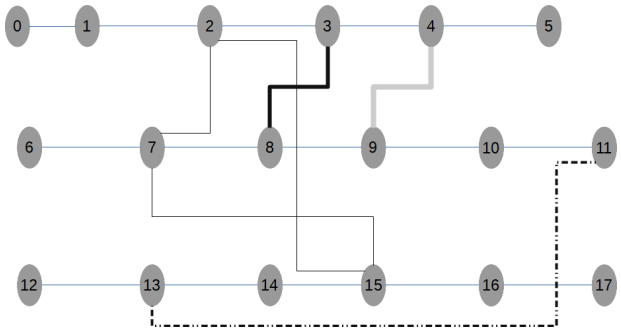

Figura 5.4: Unión de las gráficas de conectividad y Exclusividad

En algunos casos, la red se divide en subredes de punto a punto, por lo tanto la gráfica de ruteo debe de descomponer sus vértices para poder interconectar el segmento de bloque C que comparten las subredes con el CLB, es por ello, que mediante el ejemplo de la Figura 5.5, se explica el procedimiento de descomposición.

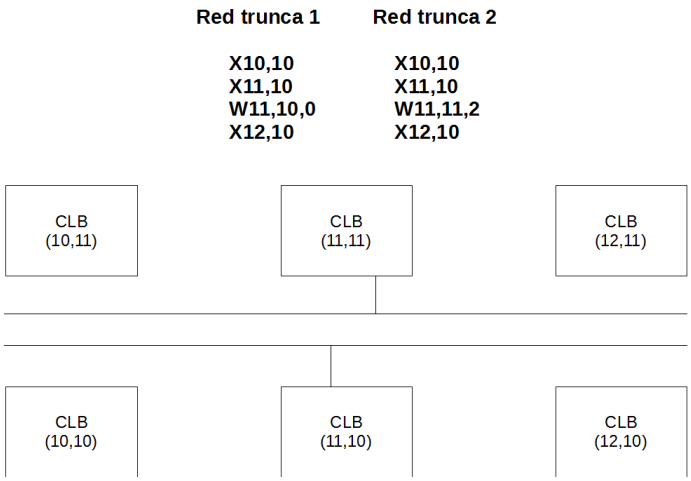

Figura 5.5: Gŕafica del ruteo global

Por cada segmento X o Y que atraviese un bloque C, se añade un vértice de tipo C a la gráfica. En el caso de que un bloque C tenga dos o más vértices de tipo C, se crea una gráfica completa dentro del bloque C, uniendo los vértices C mediante aristas de exclusividad. La gráfica que se genera en cada uno de los bloques C es una gráfica completa K_n de n vértices, como lo muestra la Figura 5.6. Este tipo de gráfica, permite asegurar la exclusividad mediante una coloración propia, es decir que para cada uno de los bloques C en donde existe una gráfica de tipo K_n, la coloración propia impide que físicamente dos señales ocupen la misma pista en un bloque C.

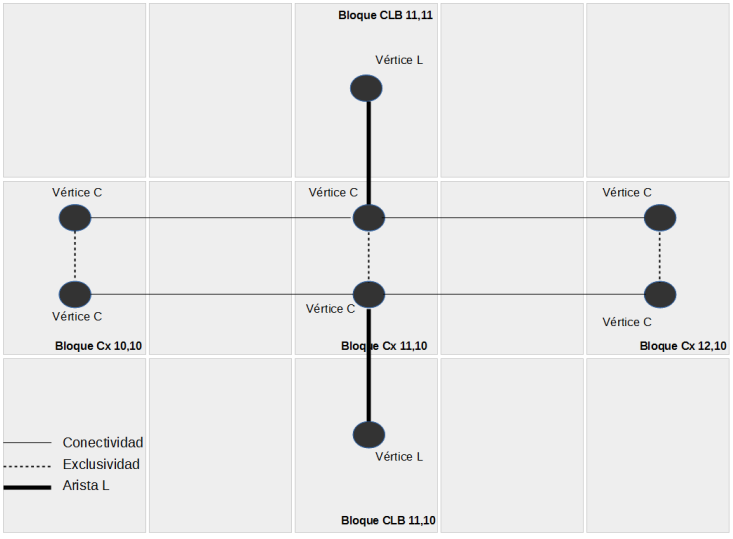

Figura 5.6: Asignación de los vértices C.

La conexión de un pin de bloque CLB hacia un bloque C, complica en gran medida la factibilidad de la gráfica de la Figura 5.6, ya que un solo pin del bloque CLB puede conectar con todas las pistas de su bloque C adjunto, por ello es necesario hacer un reordenamiento de los vértices C. Cada vértice C, perteneciente a una red que atravieza un bloque C, se duplica. Cada copia se conecta a su

correspondiente vértice L por medio de aristas tipo L. A este proceso se le llama el método de división y puede verse gráficamente en la Figura 5.7.

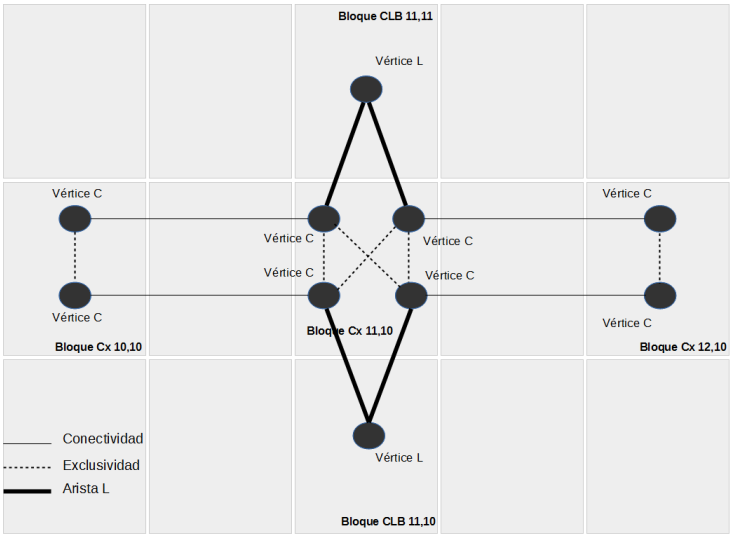

Figura 5.7: Gráfica final del ruteo global

Para resolver el problema de asignación expresado en la gráfica de conectividad-exclusividad, es necesario especificar uno de tres niveles de ruteo posibles, los cuales se listan en orden de complejidad:

1. **Nivel de redes.** Para este nivel, la gráfica resultante contiene un vértice por red, la cual es coloreada con un número de colores igual al número de pistas por canal que contenga la arquitectura seleccionada. i.e. La arquitectura del bloque S con Fs=3 impide hacer cambio de pistas, es decir si la señal llega al bloque S por la pista uno (color uno), no importando la dirección que tome, la señal debe de salir por la pista uno.

   Seleccionar el ruteo a nivel de redes, permite simplificar en gran medida la gráfica, ya que una sola red, tiene el mismo color para todos los segmentos, con ello la gráfica de conectividad puede reducirse, si algún vértice aparece dos o más veces en la gráfica. Al reducir la gráfica, el número de cálculos matrices y listas ligadas que conlleva, se reduce enormemente ya que una red, sólo aparecerá una vez en la gráfica. En la Figura 5.8, se muestra un ejemplo de una gráfica de exclusividad que tiene varios vértices que se repiten. Al contrario de los vértices que se repiten, las aristas que los conectaban siguen teniendo la misma conexión al único vértice que se contrajo.

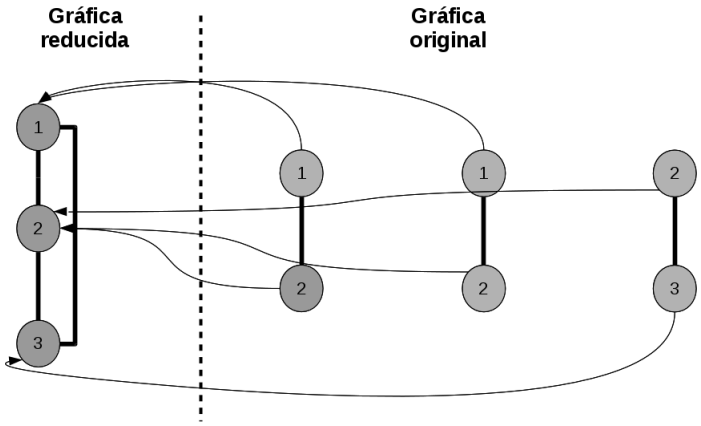

Figura 5.8: Reducción de la gráfica de exclusividad en donde los vértices son iguales

2. **Nivel de subredes.** Para este nivel, la gráfica resultante contiene un vértice por cada subred de una red, la cual es coloreada con un número de colores igual al número de pistas por canal que tenga la arquitectura seleccionada. Para un ruteo a nivel de subredes, todos los vértices de una red pueden ser del mismo color o tomar un color diferente en los vértices de cada subred. La arquitectura del bloque S es exactamente igual a la del ruteo por redes i.e. Fs=3, por lo que conserva las mismas características que el ruteo por redes.

   Es necesario hacer notar que para la gráfica a nivel de redes y subredes, la única posibilidad de rutear el circuito, es con un FPGA que en su arquitectura contemple bloques S con un valor de $Fs = 3$, es decir que la señal puede rotar a 0, 90 y 270°, pero no cambiar su pista original.

3. **Nivel de segmentos.** Para este nivel, la gráfica resultante, contiene un vértice por cada segmento que compone la red, la cual es coloreada con un número de colores igual al número de pistas por canal que tenga la arquitectura seleccionada. Para el ruteo a nivel de segmentos, cada segmento que existe en la red, puede tomar cualquiera de los colores existentes para colorear la gráfica, ya que la arquitectura del bloque S, permite conectar cualquier entrada con cualquier pista, es decir si una señal llega al bloque S por la pista uno y el canal está compuesto por tres pistas, la señal puede salir por la pista cero uno o dos respectivamente.

Capítulo 6

Heurística - Búsqueda indexada

En el estado del arte, es posible encontrar diversas técnicas de ruteo de los dispositivos lógicos programables, tales como el recocido simulado, búsqueda tabú, ruteo busca minas, búsqueda de caminos, entre otras [10, 11, 24, 25]. La gran mayoría de estas técnicas heurísticas, intentan realizar el ruteo evaluando una ruta a la vez. Si la ruta que en ese momento se está evaluando tiene conflicto, debe ser nuevamente ruteada. Esto conlleva grandes cantidades de tiempo de procesamiento para realizar el ruteo y una calidad que puede mejorarse en gran medida. Esta tesis propone codificar el problema del ruteo detallado a partir de dos gráficas, una de conectividad y otra de exclusividad, cuya coloración resolverá el problema de asignación. La búsqueda de la coloración mencionada se realiza a traves de una heurística basada en un árbol R.

6.1. Árbol R

Un árbol *R* es una estructura de datos, que permite clasificar la información de manera multidimensional similar a las características de los árboles B [22]. La única diferencia entre estos dos árboles, radica en que el árbol R almacena los datos de manera espacial. Este tipo de árbol, divide los datos en forma jerárquica en conjuntos de distintos niveles. A su vez, aprovecha el orden que estos niveles generan, para efectuar una búsqueda de una manera rápida y eficiente. En un árbol R los datos se agrupan mediante un rectángulo llamado rectángulo de mínimo enlace. En cada nodo pueden existir tantos objetos como sea necesario y los nodos hoja siempre apuntan a los objetos actuales.

Para ejemplificar el funcionamiento básico de un árbol R, se propone el problema de clasificar materias, por su grado de complejidad y por el número de horas requeridas. Los datos del Cuadro 6.1, representan materias cursadas por alumnos, asi como la complejidad que infiere cursar la materia y el número de horas obligatorias. En la medida que los datos crecen, búscar un óptimo en n dimenciones, se vuelve cada vez más complejo, por lo tanto, tener indexados los datos de manera ordenada y jerárquica, agiliza la búsqueda.

Materia	Complejidad	Número de horas
Matemáticas	10	68
Física	8	103
Química	2	32
Civismo	1	12
Español	6	43
Lógica	7	86
Geografía	5	47
Historia	6	35
Álgebra	8	46
Diseño	3	19
Bioligía	4	11
Inglés	4	48

Cuadro 6.1: Base de datos del árbol R

Para construir cualquier árbol R, se requieren dos pasos:

División este paso consiste en almacenar los datos uno por uno, hasta que la capacidad n de la hoja se desborde. En ese momento la hoja se divide en dos hojas de capacidad n cada una y la raíz se convierte en rectángulos R_i

Inserción al agregar un nuevo dato en el árbol R, hay que tener en cuenta en cuál de las siguientes tres opciones recae:

1. **En su rectángulo.** Si el dato a agregar se ubica sobre un rectángulo ya definido, entonces el dato pertenecerá al conjunto de datos de dicho rectángulo.
2. **Si entra en varios rectángulos.** Si el dato a agregar se ubica entre varios rectángulos, entonces el dato pertenecerá al conjunto de datos del rectángulo más pequeño.
3. **Si no entra en ninguno.** Si el dato a agregar se ubica en una zona en donde no existen rectángulos, entonces se toma el rectángulo que vaya crecer menos espacialmente y se modifican sus dimensiones, de tal manera que el rectángulo contenga al nuevo dato.

La Figura 6.1, es un vector que representa las materias ordenadas aleatoriamente.

Figura 6.1: Vector de materias utilizado para contruir el árbol R.

Para este ejemplo, cada nodo podrá almacenar un máximo de cinco materias. Los datos se almacenan uno por uno dentro de la raíz del árbol. Cuando ésta se desborda, se realiza el método de división mencionado anteriormente. La Figura 6.2, muestra el proceso de división para los primeros seis datos y la Figura 6.3, muestra gráficamente cómo se genera esta división.

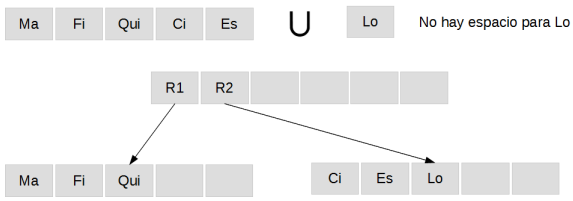

Figura 6.2: División del árbol R

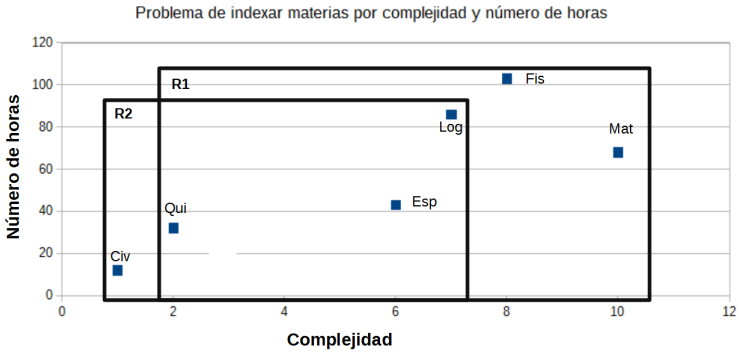

Figura 6.3: Representación gráfica de la división del árbol R.

A continuación, se agrega un dato a la vez mediante el método de inserción mencionado anteriormente (Figura 6.4), hasta que alguna hoja se desborde y sea necesario realizar una nueva división (Figura 6.5).

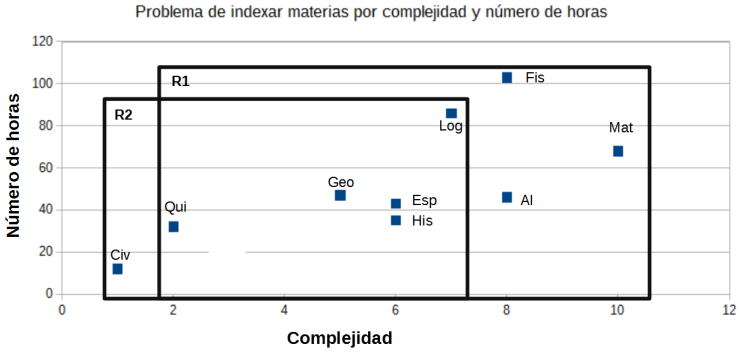

Figura 6.4: Inserción.

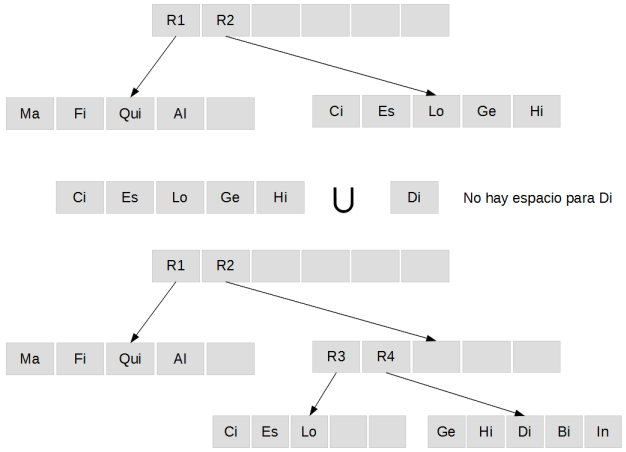

Figura 6.5: Proceso de inserción, división y árbol R.

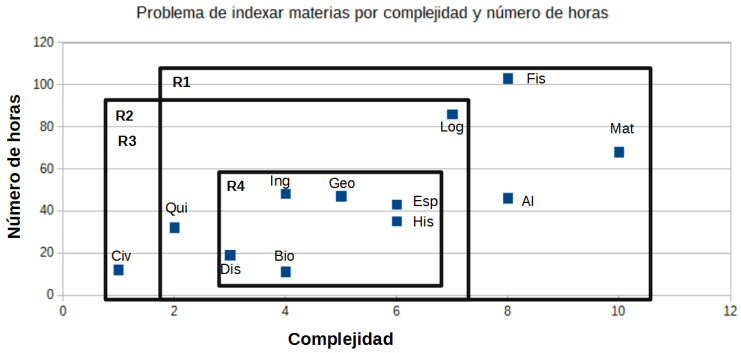

Figura 6.6: Representación gráfica del árbol R.

Algoritmo de búsqueda en el árbol R.

Para realizar una búsqueda dentro del árbol R, se desciende desde la raíz, siguiendo a los hijos cuyos rectángulos, se intercectan con el área de consulta Z, hasta llegar a las hojas del árbol.

Para el ejemplo de la Figura 6.7, el espacio de búsqueda intersecta a los rectángulos R2, R3 y R4, por lo tanto se comienza la búsqueda desde la raíz en R2 hacia los hijos R3 y R4, lo cual nos indica que los posibles datos que están dentro de esa área de búsqueda son:

civismo, **español, lógica, geografía, historia**, diseño, biología, inglés.

Por lo tanto, el árbol R, proporciona un subconjunto del conjunto de todos los datos,en donde una heurística puede encontrar con mayor facilidad los datos que se encuentran al interior del espacio de búsqueda Z.

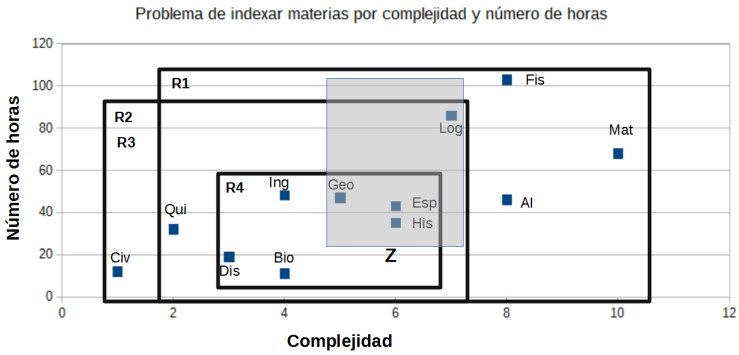

Figura 6.7: Búsqueda dentro del árbol R mediante el rectángulo Z

6.1.1. Organización del ruteo mediante el árbol R.

La teoría del árbol R, puede adaptarse fácilmente al ruteo del FPGA, en este sentido, se propone un árbol R en $\mathbb{R}^3$, en donde las coordenadas verticales, se encuentran definidas por los vértices de la gráfica conectividad-exclusividad, que se describió en el Capítulo 5. En el eje horizontal, las coordenadas hacen referencia, a los niveles de conflicto en que la función objetivo puede disminuir o aumentar en su valor actual. La gráfica resultante de estos dos ejes coordenados, es una gráfica de dispersión, en la que la intersección de sus coordenadas apunta a objetos llamados páginas. Dentro una página, se encuentran contenidos todos los colores activos que puede utilizar el vértice, es decir aquellos colores que puede tomar el vértice y que disminuyen o aumentan en n la función objetivo SAT de acuerdo al nivel de conflicto donde se ubique la página. El la Figura 6.8 se muestra un ejemplo de la estructura que tiene el árbol R propuesto, con la cual, es posible indexar los datos del ruteo global de manera ordenada.

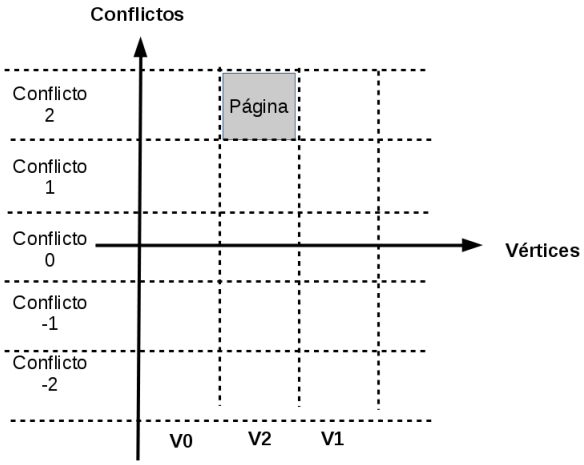

Figura 6.8: Árbol R para la estructura de datos del ruteo.

6.2. Heuristica

Para ejemplificar el modelo heurístico de búsqueda indexada de esta tesis se partirá de una gráfica simple que no contempla algunos aspectos del modelo de gráficas del Capítulo 5, pero que tienen exactamente la misma funcionalidad.

Como entrada, las instancias proporcionadas a la heurística de búsqueda indexada, provienen del VPR y se encuentran en el formato descrito en la Sección 3.3. En general el archivo de ruteo global que se proporciona, contiene la descripción de las interconexiones que se realizan entre los bloques CLB y las entradas-salidas. La descripción detallada, está separada por redes que a su vez contienen subredes y que su elemento principal de construcción son los segmentos coordenados. Es preciso mencionar que la ruta especificada por este archivo de ruteo global, no es modificada en lo absoluto por el ruteo detallado, sin embargo las pistas que toman los segmentos sobre el canal, si serán asignadas.

6.2.1. Método de intensificación dirigida

Para describir eficientemente la heurística y su relación con el árbol R, se utiliza la gráfica mostrada en la Figura 6.9 a manera de ejemplo. Dicha gráfica contiene tres vértices que inicialmente se encuentran en conflicto, en la restricción de exclusividad, y provienen de una instancia hipotética que se implementará en una arquitectura con siete pistas por canal, por lo que la gráfica se coloreará con siete colores $C = \{0, 1, \ldots, 6\}$.

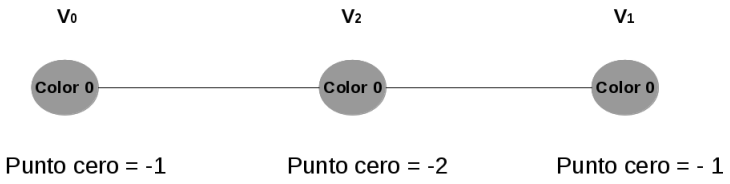

Figura 6.9: Gráfica G, mediante la cual se evaluan los puntos cero de cada vértice mediante coloración condicional

La heurística requiere evaluar los puntos en los que la función objetivo SAT descrita en la Sección 3.3, disminuye de manera considerable, por ello la inicialización de los datos comienza con un vector de vértices ordenados de manera ascendente y asigna el color cero a todos los vértices de la gráfica. Para calcular el índice que disminuya la mayor cantidad de conflictos de la función objetivo, es necesario calcular "el punto cero", el cual sirve para definir la posición de la página de un vértice seleccionado. El punto cero se calcula como el número de aristas que violan la función de restricción XNOR $\rightarrow 1$. Una vez que se tiene el cálculo del punto cero para todos los vértices de la gráfica, se escoge aquel vértice que tenga el índice del punto cero más negativo. En el ejemplo que se muestra en la Figura 6.10, el vértice dos genera dos conflictos con sus vecinos, por lo tanto, el punto cero de este vértice es -2.

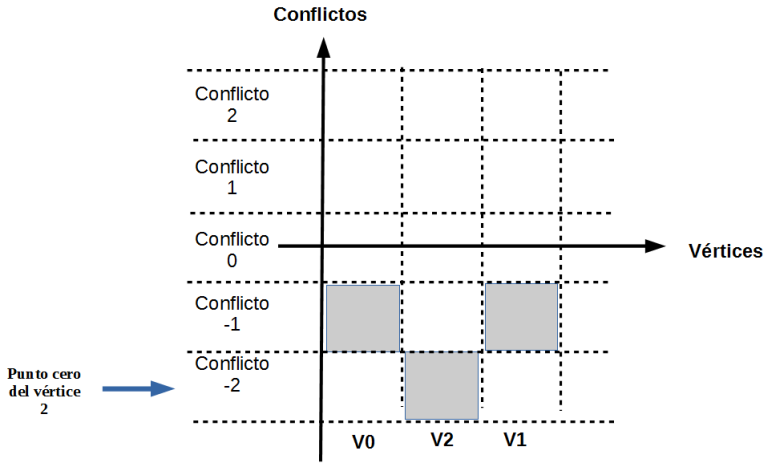

Figura 6.10: Página que contiene los colores que disminuyen la función objetivo en dos

Al observar la página del vértice dos con punto cero = -2, pueden encontrarse tres de los siete colores (pistas) que hacen que la función objetivo disminuya en dos conflictos. De los posibles colores que se tienen (1, 5, 6) siempre se escoge el que está más a la izquierda, por lo tanto el vértice dos seleccionará como nuevo color el uno. Cabe mencionar que un vértice puede tener una o más páginas. Para este ejemplo, el vértice dos podría tener una página más con los movimientos 0,2,3 y 4.

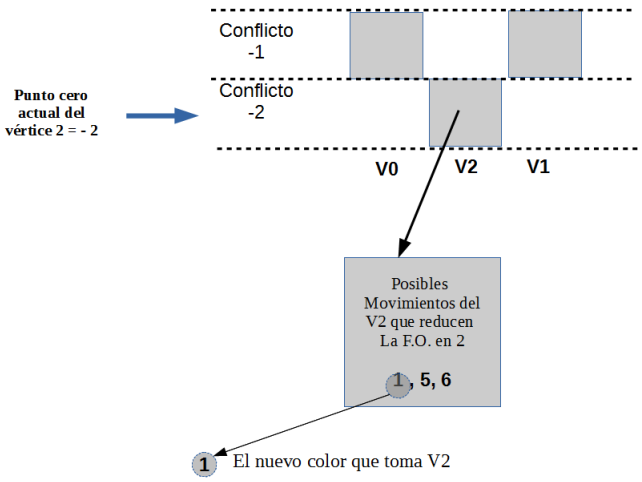

Figura 6.11

El siguiente paso, consiste en obtener el nuevo "punto cero actual" del vértice dos, con su nuevo color, evaluando a sus vecinos mediante la coloración condicional. En este caso, el nuevo color, genera un "punto cero actual" del vértice dos igual a cero. Para generar entropía, el vértice dos se desplaza en el árbol R al final de todos los vértices. Esto asegura que al llegar a un valle, se evalúen siempre los vértices que están más a la izquierda, en caso de tener un empate entre páginas.

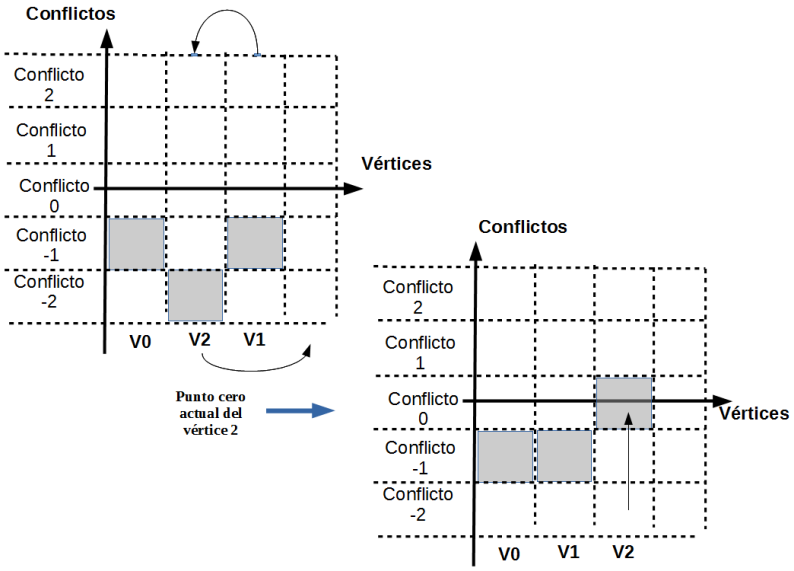

Figura 6.12: Cálculo del nuevo punto cero del vértice dos y su desplazamiento para generar entropia

La diferencia entre el punto cero actual y el punto cero anterior, desplaza todas las páginas del vértice en esa cantidad como se muestra en la Figura 6.12. Finalmente, los vecinos que interactúan con el vértice dos, se evalúan mediante coloración condicional con el nuevo color (1) y se aumenta o disminuye sus páginas en un conflicto, vease la Figura 6.13. Éste método, se repite, evaluando siempre la página con el índice de conflictos más negativo, hasta que la función objetivo sea igual a cero.

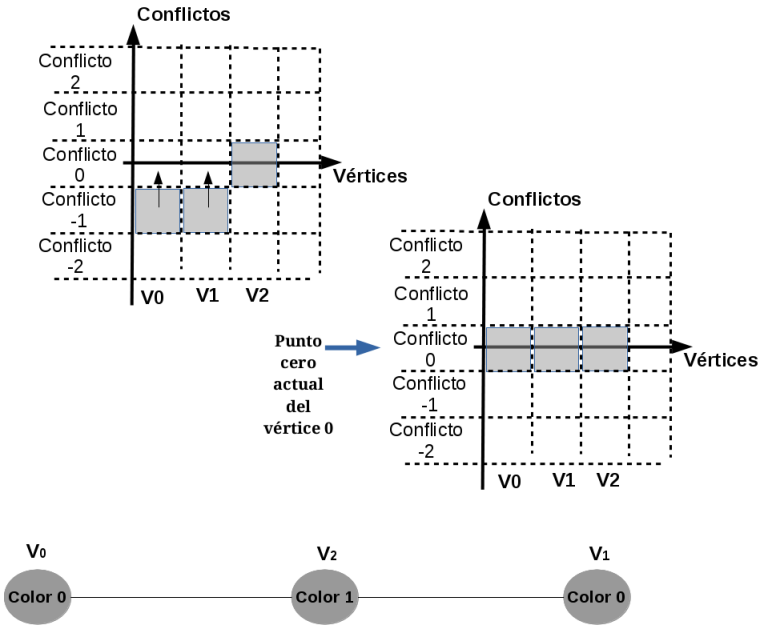

Figura 6.13

6.2.2. Método de diversificación Dirigida.

Cuando el método de intensificación ha llegado a un valle, todas las páginas se encuentran en un índice de conflicto mayor o igual a cero. Para poder escapar de este valle, se aplica un método de diversificación dirigida, que búsca maximizar la probabilidad de salir del valle, esto se logra bajando la mayor cantidad de páginas posibles, por medio de una búsqueda en la que se analizan los posibles movimientos de cada vértice (no sólo los de la izquierda), seleccionando aquel vértice en el que su movimiento baje más páginas, sin importar que la función objetivo aumente, tal y como lo muestrá la Figura 6.14. El método probabilístico mencionado anteriormente, toma en cuenta los posibles movimientos de cada vértice que afecten a la mayor cantidad de vecinos posibles.

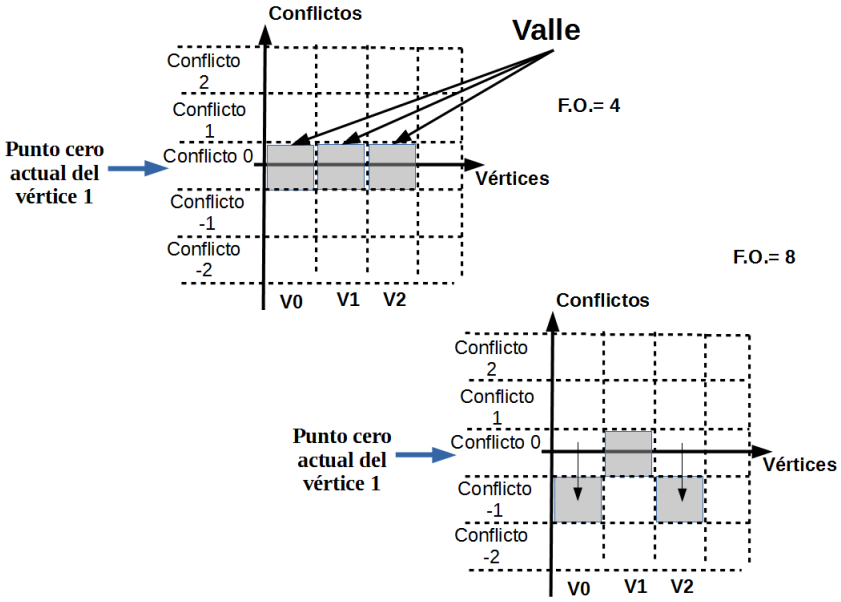

Figura 6.14: Diversificación

Capítulo 7
Resultados

En esta tesis se explicó el método heurístico, propuesto para procesar un ruteo detallado en un FPGA con diseño de islas, el cual fue optimizado para evaluar varias rutas a la vez. Esta nueva técnica que se implementó, redujo ampliamente los tiempos necesarios para obtener un ruteo detallado, en comparación a los tiempos que produce el programa VPR. Además,como los muestra el Cuadro 7.1, en dos instancias fue posible encontrar un ruteo detallado, con un menor número de pistas que el VPR. Esta reducción de tiempos y mejora en la calidad del ruteo, se debe en gran medida a tres aspectos importantes de esta tesis:

La satisfacibilidad booleana. Modelar el ruteo como un problema de satisfacibilidad booleana, trae consigo varias ventajas. Una de ella, es que a pesar de tener miles de rutas sólo dos restricciones acotan el problema. Otra gran ventaja, es que la heurística puede satisfacer varias rutas a la vez, al contrario del modelo que impera en el estado del arte "búsqueda de caminos" [25] [10] [24] que sólo intenta conectar una ruta por vez.

Coloración condicional. Fue planteada para este proyecto de investigación y establece que la resolución del problema de satisfacíbilidad booleana, se transforma en un programa que sólo tiene que resolver sumas y restas en listas ligadas, en lugar de resolver conjunciones y disyunciones en miles de cláusulas. Esto reduce el tiempo de resolución del problema en gran medida.

Árbol R. Para cualquier heurística, la etapa de inicialización, es muy importante, ya que una muy buena inicialización, podría dejar a la heurística muy cerca del óptimo global que se está buscando. Aunado a esto, si los datos que se le entregan, dejan de tener un carácter completamente aleatorio, el número de evaluaciones de la función objetivo disminuirán. Por lo anterior, un árbol R fue la mejor solución para indexar los datos que se le proporcionarón a la heurística, ya que en todo momento se sabe el tipo de movimientos que ayudarán a disminuir o aumentar la función objetivo. Una ventaja más de este tipo de árbol, es que cualquier cambio que se realice en un vértice, se ve reflejado instantáneamente en sus vértices vecinos y por lo tanto un cambio en cualquier vértice, aumentará o disminuirá en uno los conflictos de cada uno de sus vecinos.

La Figura 7.1, es una gráfica de dispersión que en su eje vertical, compara la diferencia de los tiempos obtenidos por el ruteo detallado del programa VPR y el de la búsqueda indexada. Asimismo, el eje vertical representa la distancia entre las dos soluciones. El ruteo global para cada instancia fue exactamente el mismo para los dos programas, y fueron obtenidas del VPR. Claramente, puede observarse que los tiempos obtenidos por la búsqueda indexada son mucho menores que los obtenidos por el programa VPR y además, la asignación de segmentos, comparte en promedio el 40 % de los segmentos. Las dos pruebas fueron realizadas en la misma computadora, con un procesador Intel I3 y 8GB de ram.

Para resolver el problema de ruteo detallado, se eligieron siete instancias de la MCNC "taller internacional de síntesis lógica y extensiones de las instancias de síntesis lógica y optimización de 1989". Las instancias que a continuación se enlistan, representan varios circuitos combinacionales sintetizados mediante un HDL, las cuales describen el ruteo global de las interconexiones entre bloques.

Alu4 Representa una Unidad aritmética lógica[1] y su función consiste en realizar operaciones aritméticas tales como sumas, restas, multiplicaciones, divisiones y comparaciones. El ruteo global para esta instancia fue diseñado para el modelo FPGA-4005E, que cuenta con una capacidad de 42 $\times$ 42 bloques CLB, de los cuales se ocupan 1522. El circuito cuenta con dos entradas de 4 bits cada una, un selector de función de 5 bits, 1 bit de acarreo de entrada, 1 bit de acarreo de salida y siete bits de salida. El número de redes utilizado para esta instancia es de 1541.

Seq Representa un circuito lógico secuencial, en el que la salida no sólo depende del estado presente de las entradas, sino que también depende de su estado pasado. Estos circuitos son utilizados para generar máquinas de estados finito. El ruteo global para esta instancia fue diseñado para un modelo FPGA-408E, que cuenta con una capacidad de 42 $\times$ 42 bloques CLB, de los cuales son ocupados 1750. El circuito cuenta con 41 entradas y 35 salidas. El número de redes utilizadas para esta instancia es de 1993.

Tseng Representauna memoria de acceso aleatorio[2]. El ruteo global para esta instancia fue diseñado para el modelo FPGA-4013E, que cuenta con una capacidad de 33 $\times$ 33 bloques CLB, de los cuales se ocupan 1047. El circuito cuenta con 52 entradas y 122 salidas. El número de redes utilizadas para esta instancia es de 1575.

ex5p Representa un circuito que describe una máquina de estados finitos. El ruteo global para esta instancia fue diseñado para el modelo FPGA-4013E, que cuenta con una capacidad de 33 $\times$ 33 bloques CLB, de los cuales son ocupados 1064. El circuito cuenta con 8 entradas y 16 salidas. El número de redes utilizadas para esta instancia es de 1491.

S298 Representa un controlador de semáforos viales. El ruteo global para esta instancia fue diseñado para el modelo FPGA-4025E, que cuenta con una capacidad de 44 $\times$ 44 bloques CLB, de los cuales son ocupados 1931. El circuito del cuenta con 4 entradas y 6 salidas. El número de redes utilizadas para esta instancia es de 8304.

S38417 Representa un circuito combinacional que contiene 1564 flip-flops y 4933 compuertas y es una instancia de verificación del nivel de compuertas. El ruteo global para esta instancia fue diseñado para el modelo FPGA-4030E, que cuenta con una capacidad de 81 $\times$ 81 bloques CLB, de los cuales son ocupados 6406. El circuito cuenta con 29 entradas y 106 salidas. El número de redes utilizadas para esta instancia es de 6435.

Frisc Representa un circuito que realiza una reducción rápida de un conjunto de instrucciones de computadora[3]. El ruteo global para esta instancia fue diseñado para el modelo FPGA-4035E, que cuenta con una capacidad 60 $\times$ 60 bloques CLB, de los cuales son ocupados 3556. El circuito cuenta con 20 entradas y 116 salidas. El número de redes utilizadas para esta instancia es de 3576.

[1] ALU por sus siglas en ingles Arithmetic Logic Unit
[2] RAM por sus siglas en ingles Random-access memory
[3] FRISC por sus siglas en ingles Fast Reduced Instruction Set Computer

Para obtener el número mínimo de pistas necesarias que se requieren en la arquitectura del FPGA, se ejecutó el de ruteo detallado en el VPR, con un criterio de paro que refleja cinco veces el tiempo normal de su ejecución.Para evaluar el nivél de similitud entre el ruteo detallado del VPR y el de la búsqueda indexada, se comparó en cada red la asignación de la pista para cada segmento dado, como resultado, se obtuvo un promedio de 38.28 % de similitud en sus asignaciones.

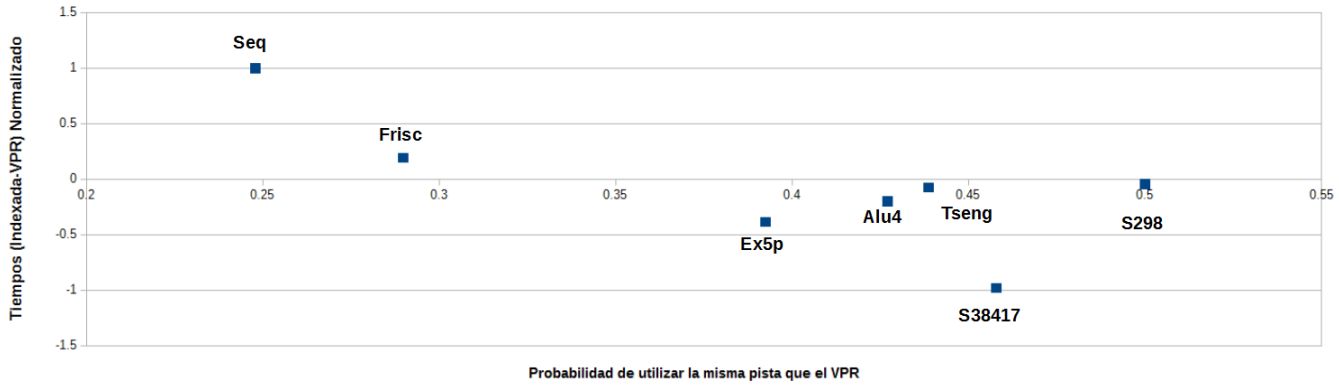

Figura 7.1: Gráfica de dispersión de la diferencia de tiempos entre el VPR y la búsqueda indexada (normalizados) y la probabilidad de realizar el mismo ruteo que el VPR

Instancia	Tiempo B. Indexada	Tiempo VPR	Ventaja porcentual	Menor número de pistas
Alu4	5.512	10.64	**48.20 %**	**Si**
Ex5p	0.063	9.916	**99.36 %**	**Si**
frisc	40.136	35.2	-14.02 %	=
s38417	8.052	33.11	**75.68 %**	=
seq	49.694	24.12	-106.03 %	=
tseng	0.038	2	**98.10 %**	=
s298	23.2	24.36	**4.76 %**	=

Cuadro 7.1: Comparación de resultados entre el VPR y la búsqueda indexada

Capítulo 8
Conclusiones

En general el ruteo detallado que proporciona el programa VPR, tiene muy buena calidad en la asignación del ruteo, sin embargo al utilizar la técnica de búsqueda de caminos en su algoritmo, tiene grandes posibilidades de estancarse en óptimos locales a medida que el número de redes aumenta. Esto se debe a que toma como base, un primer ruteo detallado completamente congestionado e intenta mejorarlo en cada iteración. Por lo tanto, si el primer ruteo detallado congestionado es de mala calidad, es muy probable que las siguientes iteraciones también lo sean. La búsqueda indexada propuesta en este proyecto, resolvió los problemas mencionados anteriormente, ya que en la estructura de su ruteo, toma en cuenta el siguiente aspecto: Para evitar el problema de tratar una sola ruta a la vez como lo hace el programa VPR, el modelo de satisfacibilidad booleana, crea el ambiente necesario para tratar el ruteo de varias redes a la vez, lo cual aceleró en gran medida el tiempo de procesamiento que este ruteo requiere. La coloración condicional, ayudó a que las cláusulas conjuntivas y disyuntivas de la ecuación de satisfacibilidad booleana, se transformarán en simples sumas y restas, con ello, la implementación no tiene que evaluar cláusulas condicionales. El árbol R, potenciabilizó a la heurística, ya que todos los movimientos están indexados de una manera completamente ordenada y ligada. Esto da como resultado, que la búsqueda se realice en un tiempo mínimo y que ademas, en algunos casos, se encuentre un ruteo que puede realizarse con menos pistas.

Durante esta investigación también se utilizó un algoritmo genético con codificación real para intentar resolver el problema de satisfacibilidad. Sin embargo, no se obtuvieron resultados consistentes incluso en su versión paralela, debido a que la técnica de cruza utilizada (cruza de dos puntos), no contribuía a mejorar la solución SAT de una manera eficiente.

8.1. Trabajo futuro

La heurística de búsqueda indexada que se presentó en este trabajo, depende en gran medida del ruteo global proporcionado por el programa VPR, sin embargo, un trabajo futuro, podría contemplar las tres etapas del ruteo (colocación, ruteo global y ruteo detallado) en un solo problema. Como hipótesis, esto podría realizarse mediante alguna técnica de optimización multi objetivo, por ejemplo, la heurística MOEA/D [26]. De esta manera, se aseguraría que el objetivo de cada una de estas tres etapas, sea óptimo en conjunto.

Bibliografía

[1] Debapriya Basu Roy, Shivam Bhasin, Sylvain Guilley, Jean-Luc Danger, Debdeep Mukhopadhyay, Xuan Thuy Ngo, and Zakaria Najm, "Reconfigurable LUT: A Double Edged Sword for Security-Critical Applications", Security, Privacy, and Applied Cryptography Engineering, paginas 248-268, 2015, Springer International Publishing.

[2] Larry McMurchie, Guest Lecture: Placement and Routing for FPGAs, Synopsys Inc. Formerly with Depts. of EE and CS, UW. [web; accesado el 13-Feb-2016].

[3] Chandra Mulpuri and Scott Hauck. 2001. Runtime and quality tradeoffs in FPGA placement and routing. In Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays (FPGA '01), Martine Schlag and Russell Tessier (Eds.). ACM, New York, NY, USA, 29-36.

[4] Jiang Hu and Sachin S. Sapatnekar, A Survey on Multi-Net Global Routing for Integrated Circuits, Integration, the VLSI Journal, 2001, volumen 31, páginas 1-49.

[5] Abboud,N., Grötschel, M., Koch, T., Mathematical methods for physical layout of printed circuit boards: an overview. OR Spectr. 30(3), 453-468 (2008).

[6] Stephen A. Cook. 1971. The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing (STOC '71). ACM, New York, NY, USA, 151-158.

[7] Christos H. Papadimitriou. 2003. Computational complexity. In Encyclopedia of Computer Science (4th ed.), Anthony Ralston, Edwin D. Reilly, and David Hemmendinger (Eds.). John Wiley and Sons Ltd., Chichester, UK 260-265.

[8] Taghavi T., Ghiasi S. and Sarrafzadeh M., "Routing Algorithms: Architecture-Driven Rerouting Enhancement for FPGAs," In Proc. Of IEEE International Symposium on Circuits and Systems (ISCAS), 2006.

[9] Eugenio Carnero García, "Evolución Tecnológica del Hardware Xilinx (1985-2015).", Manual de referencia, Universidad Autónoma de Madrid, Departamento de Tecnología Electrónica y de las Comunicaciones, Julio 2015.

[10] M. Hutton and V. Betz, "FPGA Synthesis and Physical Design,"Volume 1, Chapter 13, in Electronic Design Automation for Integrated Circuits Handbook, L. Scheffer, L. Lavagno, and G. Martin, Eds., Taylor and Francis CRC Press, 2006.

[11] Nam, G.-J., Aloul, F., Sakallah, K., and Rutenbar, R., "A Comparative Study of TwoBoolean Formulations of FPGA Detailed Routing Constraints," in Proc. Int. Symp. on Physical Design, 2001.

[12] Antonin Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", Proc. ACM-SIGMOD International Conference on Management of Data, Boston, Mass., 1984.

[13] Dario Coltorti and Andrea E. Rizzoli. 2007. Ant colony optimization for real-world vehicle routing problems. SIGEVOlution 2, 2 (July 2007), 2-9.

[14] P. Wang, R. Krishnamurti, and K. Gupta. View planning problem with combined view and traveling costs: Problem formulation, hardness of approximation, and approximation algorithms. Technical Report TR2006-17, Simon Fraser University, Burnaby, B.C., Canada, May 2006. también existe una versión publicada por ICRA 2007.

[15] Ian Kuon, Russell Tessier, and Jonathan Rose. FPGA Architecture: Survey and Challenges. In Foundations and Trends in Electronic Design Automation, Volume 2, Number 2, pp. 135–253, 2007.

[16] Adrian Ludwin and Vaughn Betz. 2011. Efficient and Deterministic Parallel Placement for FPGAs. ACM Trans. Des. Autom. Electron. Syst. 16, 3, Article 22 (June 2011), 23 pages.

[17] Nobert, Y. and Picard, J.-C. (1996), An optimal algorithm for the mixed Chinese postman problem. Networks, 27: 95–108.

[18] J. Soukup, "Circuits Layout", Proc. of the IEEE, Vol. 69, No. 10, pp. 1281-1304, October 1981.

[19] M. J. Lorenzetti., D. S. Baeder. Routing. en Bryan Preas y Michael Lorenzetti, editores, Physical Design Automation of VLSI Systems, capítulo 3, 157-210. The Benjamin Cummings Publishing Company, Menlo Park, CA,1988.

[20] P. Alfke, 20 Years of FPGA Evolution – from Glue Logic to Major System Component,HOT CHIPS 19, Aug. 2007.

[21] Vinay Chopra and Amardeep Singh.ÂAnt Colony Optimization approach for solving FPGA routing with minimum channel widthÏInternational Journal on Computer Science and Engineering, 2011

[22] Joyanes Aguilar Luis, Algoritmos y Estructuras de Datos Una perspectiva en C, Mc Graw Hill, 2004

[23] Gi-Joon Nam, K. A. Sakallah, and R. A. Rutenbar. 2006. A new FPGA detailed routing approach via search-based Boolean satisfiability. Trans. Comp.-Aided Des. Integ. Cir. Sys. 21, 6 (November 2006), 674-684.

[24] V. Betz and J. Rose, "Vpr: A new packing, placement and routing tool for fpga research," in Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications, 1997

[25] L. McMurchie and C. Ebeling. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In International Symposium on Field Programmable Gate Arrays, Monterey, Ca., Feb. 1995.

[26] Zhang, Q., Li, H.: "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition." IEEE Trans. Evolutionary Computation, 2007.